

Neural Network and Applications
Prof. S. Sengupta
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 14
Linear Least Square Filters

Our topic for today on the chapter of single layer perceptron is, Linear Least Square Filters. But, before we discuss about this, we have to complete our unfinished discussions related to the, earlier the unconstrained optimization approach. That is the Gauss Newton method. We had gone up to some expressions, but before we start from the point where we have left yesterday. We need to also discuss about some of the clarifications, because at the end of the class when I was talking to the students, I could realize that there are few things which required clarifications.

Now, 1 of the clarification is related to the error expression that we had put forward. And some of the students were interested to know that what is the significance of that error expression, so let me go about that. See what we have done is that we are taking n number of observations. So, with a fix set of weight we are keeping the weight fixed and then, we are computing the error for n number of observations.

And those errors we are designating by e_1, e_2, e_3 etcetera up to e_n and then, we are defining the cost function as summation of half of e_i squares. So, where i is summed up from 1 to n which is the number of observations. And then, we are our objective will be to minimize this half sum of e_i square or minimizing e_i square summation in fact. So, to do that what we did was, that we had derived an error expression.

So that with the change of weight, because what we are doing is that at the end of n observations, we are going to update the weight. And the thing is that once you update the weight, then the errors that you had obtained for the n observations, those errors also change, because with change of weight, error is a function of the weights. So, when we change the weights that time the error also changes, so the error gets updated.

So, what we are doing is that, we are operating the a whole thing, we are making a search at the point W is equal to W_n , because weights are going to change.

(Refer Slide Time: 03:37)

The whiteboard contains the following handwritten text:

- Top left: $\vec{w} = \vec{w}(n)$
- Top right: A small box containing "© CET I.I.T. KGP"
- Center:
$$\underline{e'(i, \vec{w})} = \underline{e(i)} + \left[\frac{\partial e_i}{\partial \vec{w}} \right]_{\vec{w} = \vec{w}(n)} (\vec{w} - \vec{w}(n))$$
- Below the center: $i = 1, 2, \dots, n.$
- Bottom:
$$\vec{w}(n+1) = \arg \min_{\vec{w}} \left\{ \frac{1}{2} \sum \| \underline{e'(i, \vec{w})} \|^2 \right\}$$

So, at this point W is equal to W_n , when we are going to find out a new weight, there we are finding out that with change of the weight that takes place, change of weight is definitely going to be W minus W_n . Where W is going to be the new weight and W_n going to be the older weight. So, with W minus W_n being the incremental weight the new expression for the error that we are going to get for the, we are going to get new error components for each of the observations, that we already had.

So, let us say that for the i th observation, the updated error that we must have is e which will be expressible as a function of i and W . So, i means that here we are considering the i th weight and then the W which is the weight and this is equal to e_i plus $\frac{\partial e_i}{\partial W}$ W vector. And this will be evaluated at W equal to W_n , because that is the point from which we are going to change the or update the weight W minus W_n . So, this we have to update for all these i 's, all these observation we have to update the error expression.

So, this is the updated error expression and then, what we are going to do is that for the $n+1$ th weight we must put forward the minimum value, we must find out the minimum value of this expression half of e' n W vector norm of this square. And the W vector corresponding to which it gives the minimum, that W vector we are choosing as the updated weight.

So, our objective is to minimize this sum of the squares expressions, so this is what we were doing. So, those who had any doubt about the expression that we had written, I

think we will clarified now that, this is change of the weight that we are making. And then accordingly, since we are changing the weight we also need to change the error expressions that we had obtained. So, e_i was the old error and e_i^W that we have written on the left hand side is the updated weight, followed.

So, that is clarification number 1, that some the students wanted and the second clarification is related to the multiplication that we had carried out. In fact, the expression that we had obtained for this equation only.

(Refer Slide Time: 06:38)

The image shows a whiteboard with handwritten mathematical derivations. At the top right, there is a small logo for '© GET I.I.T. KGP'. The main derivation starts with the equation:

$$\vec{e}(n, \vec{W}) = \vec{e}(n) + \vec{J}(n) (\vec{W} - \vec{W}(n))$$

Below this equation, the terms are labeled: $\vec{e}(n, \vec{W})$ is labeled 'C', $\vec{e}(n)$ is labeled 'A', and $\vec{J}(n)$ is labeled 'B'. To the right of the equation, there is a vertical double bar and the expression $\|\vec{A}\|^2$.

Below the first equation, the following equations are written:

$$\vec{C} = \vec{A} + \vec{B} \quad n\text{-dim. vectors.}$$

$$\vec{C}^T = (\vec{A} + \vec{B})^T = \vec{A}^T + \vec{B}^T$$

$$\vec{C}^T \cdot \vec{C} = (\vec{A}^T + \vec{B}^T) \cdot (\vec{A} + \vec{B})$$

$$= \vec{A}^T \cdot \vec{A} + \vec{A}^T \cdot \vec{B} + \vec{B}^T \cdot \vec{A} + \vec{B}^T \cdot \vec{B}$$

The terms $\vec{A}^T \cdot \vec{B}$ and $\vec{B}^T \cdot \vec{A}$ in the final equation are underlined.

The expression that we had obtained in the matrix form was written as e_n^W equal to e_n which is the older error plus J_n , in fact it is the J_n which directly gives you the Jacobian matrix. And it signifies the partial derivatives or the derivative with respect to W vector, which in term in the matrix it translates to the partial derivative with respect to all the different components of the W 's. And then it is J_n multiplied by the change in weight which is nothing but, $W - W_n$.

So, this is what we had obtain and in fact, let us say that this is equal to the C , this is equal to A that is this one and this is equal to B . So, we are having C is equal to A plus B , where all these things C A and B these are vectors mind you, these are all n dimensional vectors. So, then any doubts from the students no, so in that case C vector transpose will be equal to A vector plus B vectors transpose which will be nothing but, A vector transpose plus B vector transpose.

So, when we are going to compute C vector transpose time C vector, the dot product of these two, if you are going to take, then it means that we have to take A vector transpose plus B vector transpose multiplied by A vector plus B vector the dot product of this. So, if you expand this, then what happens is A vector transpose A vector plus A vector transpose B vector plus B vector transpose A vector plus B vector transpose B vector.

Now, that means to say that we get four expressions, in fact that is what people are pointing out to me that we defiantly require four expressions here. One is A by A which ultimately will turn out to be norm of A square, this will be turn out to be norm of A square. This will turn out to the norm of B square in fact, this B itself is a product these two, so B that we are writing is already a dot product of J n and W minus W n. So, J n is actually a matrix, this is an n by m Jacobian matrix and this one is m by 1 vector.

So, in effect this becomes an n dimensional vector, in fact all these things, each of these things are m dimensional vectors, so not m, but n dimensional vectors. Now, the question is about this these two terms, these two cross terms that is A transpose B and B transpose A. Now, these being vectors A vector and B vector beings vectors, they are all going to be equal, both of them are going to be equal. Why, because A transpose B means what, A transpose B simply means...

(Refer Slide Time: 10:13)

The image shows a whiteboard with handwritten mathematical equations. In the top right corner, there is a small logo that reads "© CET I.I.T. KGP". The equations are as follows:

$$\vec{A}^T \cdot \vec{B} = \sum_{i=1}^n a_i b_i$$

$$\vec{B}^T \cdot \vec{A} = \sum_{i=1}^n a_i b_i$$

$$\frac{1}{2} \vec{C}^T \cdot \vec{C} = \frac{1}{2} \vec{A}^T \cdot \vec{A} + \vec{A}^T \cdot \vec{B} + \vec{B}^T \cdot \vec{B}$$

$$\frac{1}{2} \|\vec{C}\|^2 = \frac{1}{2} \|\vec{A}\|^2 + \vec{A}^T \cdot \vec{B} + \vec{B}^T \cdot \vec{B}$$

If it is n dimensional vectors, let say in that case this can be written as summation i is equal to 1 to n, because this is in dimension. And if A vector composes of the terms a i b

a_1, a_2 etcetera up to a_n and B vector consists of b_1, b_2 up to b_n , then it will be summation $a_i b_i$ is equal to 1 to n . And similarly, $B^T A$ is definition also is this, $\sum_{i=1}^n a_i b_i$, $\sum_{i=1}^n b_i a_i$ that is same in this case, so these both things are equal.

And that is why whenever you are trying to get a C transpose C vector, that is equal to $A^T A$ vector plus 2 times you can write $A^T B$ vector plus $B^T A$ vector. So, that is how we arrive at two terms, in fact what we are doing is that, we are multiplying everything by half. So that, we get here half term that means, to say that here we will be getting half norm of C square and here, we will be getting half norm of A square.

And here we will be getting this term, there is no half associated with it, because it was already multiplied by 2, so when half comes in it becomes multiplied by 1. So, here this term remains as the $A^T B$ and then, this term is $B^T A$ which should have been a norm of B square, but because this itself is a multiplication. So, this itself is $\frac{1}{2} (A^T B + B^T A)$, so what we are going to write as $B^T A$ is $W - W^T$, so that is what we are going to get.

So, $B^T A$ also will be written in that form, so that in effect we are going to get three expressions, so that was a doubt with some people had that. We are multiplying these two, but then instead of four terms we are getting three terms, but if you are looking carefully this term that you are getting, $A^T B$ here the coefficient is 1. That means, to say that those two equal terms have been combined.

So, that is why it leads to when we substitute for C , A and B , we substitute the expressions that we had considered. Because, for C we have to put forward this A equal to put forward this and B equal to this. If we do that then, we are getting this expression very easily, in fact I would like all of you to verify this yourself and feel convinced about it.

(Refer Slide Time: 13:09)

The whiteboard shows the following mathematical expressions:

$$\frac{1}{2} \|\vec{e}(\mathbf{n}, \vec{w})\|^2 = \frac{1}{2} \|\vec{e}(\mathbf{n})\|^2$$

$$+ \vec{e}^T \cdot \vec{J}(\mathbf{n}) (\vec{w} - \vec{w}(\mathbf{n}))$$

$$+ \frac{1}{2} (\vec{w} - \vec{w}(\mathbf{n}))^T \frac{\vec{J}^T(\mathbf{n}) \vec{J}(\mathbf{n})}{(\vec{w} - \vec{w}(\mathbf{n}))}$$

Below these, the partial derivative is indicated:

$$\frac{\partial}{\partial \vec{w}}$$

At the bottom, there are two vector diagrams illustrating the relationship between vectors \vec{x} , \vec{w} , \vec{y} , and \vec{r} :

$$\frac{1}{2} \vec{y}^T \cdot \vec{R} \vec{y} \quad \vec{x}^T \cdot \vec{w}$$

$$\vec{r} \cdot \vec{y} \quad \vec{x}$$

So, this is norm of e in W square is equal to half norm of e in square plus, the second term is going to be e transpose J in W minus W in this is very clear, this is may be A transpose B simply. And then, the B transpose B and B transpose B in this case is going to be half of W minus W in, this whole transpose J transpose n J in W minus W in. So, this is the expression that we are getting for B transpose B , so this is the term that we had obtained.

Now, what we are doing this whole thing we are now differentiating with respect to W vector. So, the result is pretty simple, now here all these things mind you will be differentiated, so this is actually a scalar expression. And this we are a multiplying, this we are differentiating with respect to W and then, what happens is that this term is there. And then this expression is going to be equal to 0, then whenever you are differentiating this with respect to W vector, then it is a differentiation of this form.

X transpose W vector, this is an expression of this form the second term to say, for which we already know that the result of the differentiation is X . So, that is why if you are taking this to be a transpose form of expression. Then whenever you are differentiating with respect to W then, what is it that you are going to get, you are going to get J transpose e very correct, J transpose e is what we you are going to get.

And then, here what is said that you are going to get out of this term, you are this thing if you look at it, this thing is a quadratic form of expression. So, we are having W minus W in, let us call it something, let us call it as y . So, half of y transpose and this one we call as

R, the R matrix it becomes, because here this is going to be an m by n matrix and J n is going to be n by m matrix. So, in effect it is n by m R matrix that it becomes and then, this is becomes y. So, this is a quadratic form of expression for which the derivative is going to be R y.

So, that is what we are going to write and then, this will be equated simply to 0, because what we are doing is that we are going to minimize this expression. So, we are finding out the derivate with respect to W and equating the derivate with respect to W to 0, this is what we are getting.

(Refer Slide Time: 16:56)

$$\begin{aligned} \vec{J}^T(n) \vec{e}(n) + \vec{J}^T(n) \vec{J}(n) (\vec{w} - \vec{w}(n)) &= \vec{0} \\ \vec{J}^T(n) \vec{J}(n) (\vec{w} - \vec{w}(n)) &= -\vec{J}^T(n) \vec{e}(n) \\ \vec{w} - \vec{w}(n) &= -(\vec{J}^T(n) \vec{J}(n))^{-1} \vec{J}^T(n) \vec{e}(n) \\ \vec{w}(n+1) &= \vec{w}(n) - (\vec{J}^T(n) \vec{J}(n))^{-1} \vec{J}^T(n) \vec{e}(n) \end{aligned}$$

Gauss-Newton method in its pure form.

J transpose n e n plus J transpose n J n W minus W n this is equal to the 0 vector, we are going to get that how are we getting J transpose e n, J transpose e n simply we are obtaining from the second term. And what we are obtaining from this, this is this R y form, so R y form means we are going to get this is the R, so we are writing it directly J transpose J. And then this is our y, so y also we are writing this whole thing should be equal to the 0 vector.

So, that means, to say that if we now take this on the right hand side, then we are going to get J transpose n J n multiplied by W minus W n, that is going to be minus of J transpose n e n. Now, if we pre-multiply the left hand side and the right hand side by the inverse of this product matrix. Now, this is a square matrix and let us assume that it is

non-singular. So, what we going to do is that, we are going to take inverse of this product matrix $J^T J$ product matrix, we pre-multiply by that.

So, that will give us $W^{n+1} - W^n = (J^T J)^{-1} J^T e^n$. Just multiplying the left hand side and the right hand side by this matrix terms inverse.

Student: ((Refer Time: 19:27))

Minus sign very good, so people have rightly pointed out that there is a minus sign over here. So, that is why the expression that we are getting for the updated W , instead of calling it as W we are now going to call it as W^{n+1} , because this is the updated W . So, $W^{n+1} = W^n - (J^T J)^{-1} J^T e^n$, so this is a very important relation that we have got. And this is the pure form of Gauss-Newton method, so this is the Gauss-Newton method in its pure form.

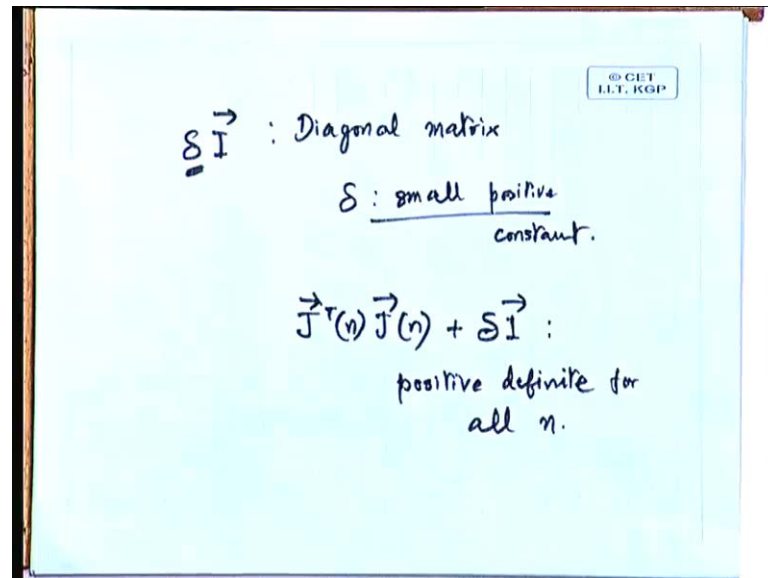
Now, that means, to say that using this expression it should be possible for us to calculate the updated weights. And all we need to know now is that, the knowledge of the Jacobian matrix, this must be known to us. Because, only then we will be able to compute this, but is the computation of this always guaranteed. Because, we have in this case made an inherent assumptions that $J^T J$ is going to be a non-singular matrix and that is why its inverse exists.

And with that assumption we have to proceed it, now one thing which can be shown that this $J^T J$ matrix that we are getting here all these are matrices. So, this $J^T J$ that we are getting it is definitely a non-negative definite, it is $J^T J$ is non-negative definite always, but there is no heard an first guarantee about non-singularity. In fact, it will be non-singular only when this $J^T J$ is having a rank equal to n , rank equal to its row that is n .

So, if this $J^T J$ happens to be rank deficient, then $J^T J$ could be singular and in which case the inverse will not exist. So, it is not always guarantee that for every n , there is no guarantee that $J^T J$ is going to be non-singular. So, there is a risk that it becomes $J^T J$ becomes rank deficient and what we need to do is to modify this slightly. So, that the rank deficiency is taken care of.

So, in order to correct for the rank deficiency what we do is that, we simply add some something to this J matrix, this product matrix that is J transpose J. So, to that we add a diagonal matrix which is of the form delta I.

(Refer Slide Time: 23:00)

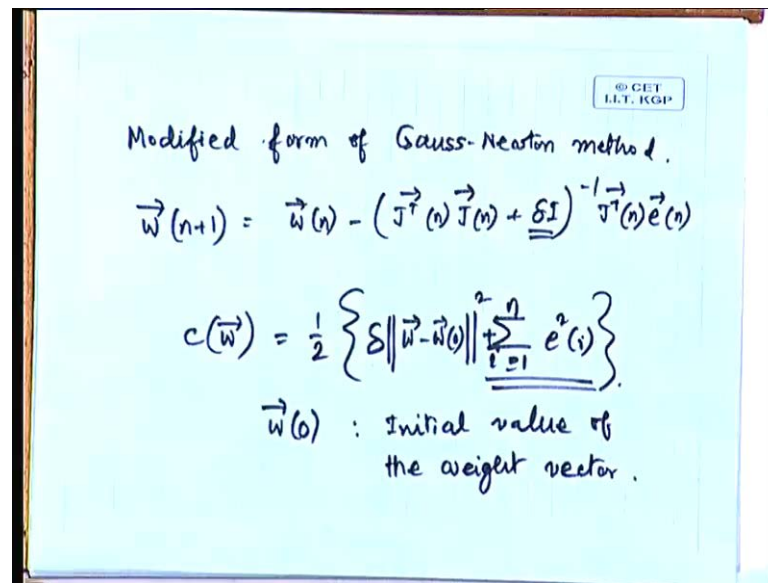


I being the identity matrix and we are multiplying that identity matrix by a constant term delta, so delta actually we have to choose delta as a small positive constant. So, delta is a small positive constant, so in this case delta I becomes a diagonal matrix, so what we are going to do is in order to correct for the rank deficiency of J transpose n J n. We are adding this diagonal matrix to that, so instead of taking J transpose J, we are going to take J transpose J plus delta I.

And in fact, the small positive constant is chosen, so that this J transpose plus delta I is ensured to be positive definite for all n. So, if we can choose a small positive constant such that, this J transpose J plus delta I is going to be positive definite for all n in that case our job is done. So, what we have to do is that, if this remains as positive deficient, then simply in this expression that we had got for the Gauss-Newton method.

All that we need to do is this term, J transpose J which should be replaced by J transpose J plus delta I.

(Refer Slide Time: 24:50)



© GET
I.I.T. KGP

Modified form of Gauss-Newton method.

$$\vec{w}(n+1) = \vec{w}(n) - \left(\vec{J}^T(n) \vec{J}(n) + \delta I \right)^{-1} \vec{J}^T(n) \vec{e}(n)$$
$$C(\vec{w}) = \frac{1}{2} \left\{ \delta \|\vec{w} - \vec{w}(0)\|^2 + \sum_{i=1}^n e^2(i) \right\}$$

$\vec{w}(0)$: Initial value of the weight vector.

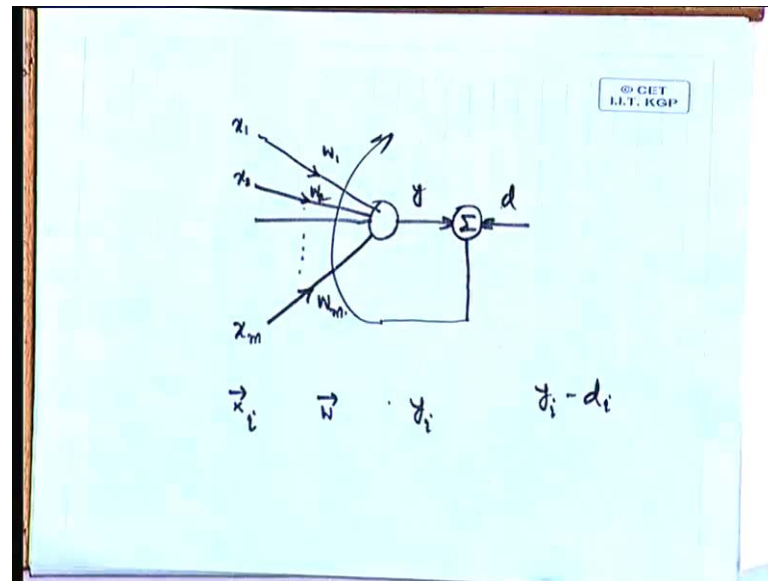
So, if we do that, then this is the expression that we have getting as the modified form of Gauss-Newton method which is $\vec{w}(n+1) = \vec{w}(n) - \left(\vec{J}^T(n) \vec{J}(n) + \delta I \right)^{-1} \vec{J}^T(n) \vec{e}(n)$. So this modification that we have done effectively is a solution of this equation. The cost function $C(\vec{w})$, actually we were earlier solving C is equal to half of summation i is equal to 1 to n $e^2(i)$, this is what we were solving earlier. But, now what happens is that it becomes say, because of the introduction of this δI , it becomes a solution of δ times norm of $\vec{w} - \vec{w}(0)$ where $\vec{w}(0)$ is going to be the initial value of the weight vector. So, this square plus this term, plus this sum of the square errors. So, effectively it is a minimization of this expression where $\vec{w}(0)$ is the initial value of the weight vector. So, here what you see is that as we increase the value of n , then this term that is sum of $e^2(i)$ this will be more and more dominant as compare to this one.

So, the effect of this could be neglected as you go in for larger and larger values of n , so this method is applicable. And now with the knowledge of Gauss-Newton method we have learnt how many three optimization techniques. Number 1 is the simple steepest descent method, number 2 is the Newton, so simple steepest descent method was nothing but, the first order approximation of the Taylor series expansion.

Then, the second optimization approach that we had studied is Newton's method, which made use of the second order terms of that Taylor series expansion, that is minimization.

And now we come to the Gauss-Newton method where as the cost objective function, we are keeping the sum of errors square and we are minimizing that. So, this is what we have got as the optimization tool and the reason, why we have got all this optimization tools with our disposal, is that we are ultimately going to use the perceptron as a linear adaptive filtering problem. In fact, this quite simple to understand.

(Refer Slide Time: 28:26)



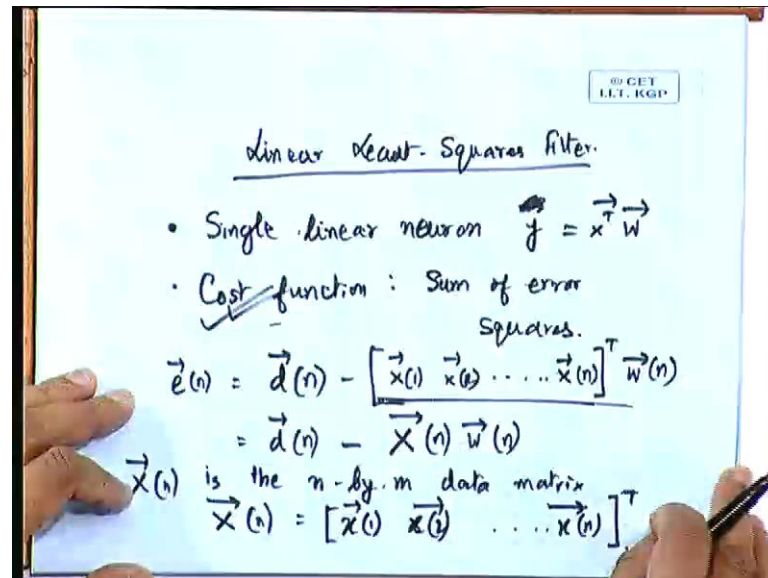
You see simple perceptron model would look like this, that say this is the neuron and we are going to have an m dimensional input, so these are nothing but, x_1 x_2 up to x_m forming an x vector. And then, we are going to have the weights as w_1 w_2 up to w_m , we are going to call that as the weight vector the input as the X vector and then, we are going to have an output of this which will be in the form of y .

So, y is going to be nothing but, either you write as X transpose y or you instance was W or you write as W transpose X whatever and then, this will be compared a against some d . So, there will be a comparison that we are going to get between the d and y , d is the desire response. So, if I take any vector x_i and we obtain y_i , then the error that is going to be y_i minus d_i or d_i minus w_i . And then, this error that is there is going to change all these set of weights again.

So, this system ultimately adapts with the change of weights, the system will be adapted to give the desire the actual output to be as slows as possible to the desired output; doing the errors minimization. So, that is why this can be looked upon as a linear adaptive filter

in problem. And in this linear adaptive filtering problem, the kind of functions that we are going to use are pretty simple one. So, in fact now we are going to study about the linear least square filter approach.

(Refer Slide Time: 30:37)



So, linear least square filter, in fact what we are going to do is pretty simple, we are going to consider two things first is a single linear neuron. Following our perceptron model we are going to take a single linear neuron that works, according to the equation y is equal to x transpose W or W transpose X whatever. And in the cost function of that you use the sum of error squares, as we have taken just now.

Now, we have already taken the sum of error squares in our Gauss Newton approach discussion, but we did not use their any single linear neuron, because we were only considering the cost function optimization approach. Earlier our emphasis was only on the cost function optimization, we temporarily forgot about the neuron network. But, when we are designing a linear least square filter, we are going to consider a single linear neurons.

So, where we are going to make use of this expression, because it is linear. So, now in this case, so linearly square filter we will use both the single linear neuron, as well as a cost function of this nature. Now, the errors vector e_n in this case can be simply represented as what as d_n , where n again is the number of observations. So, errors for

the n observations, the n th observation is equal to d_n minus and or in fact, we are writing directly in terms of vector only.

So, e_n vector which will include e_1, e_2, e_3 up to e_n which is equal to d_n vector minus this will be X_1 vector X_2 vector up to X_n vector transpose of this times W_n , so this is understood. So, d_n again is the desired response vector, so this we are going to write in the form of d_n minus we are introducing X matrix over here, $X_n W_n$ where the definition of this X matrix is this X_n is the n by m data matrix.

It is the n by m data matrix defined as X_n will be nothing but, X_1 vector X_2 vector, so on up to X_n vector transpose means this one. So, m by n m by n means what here each of these, so if you take the transpose of this transpose of this means in the column, we will be getting X_1 vector X_2 vector like that. And each of these vectors is going to be an m dimensional vector, there will be n such vectors in the row.

So, there will be n number of rows and the columns will encompass the elements of each of the vector, so since there are m elements, so there will be n rows and m columns. So, X_n is going to be a complete data matrix, that contains all the observations, all the observations are contained into it. And then, d_n is going to be simply the data, the desired responses, for each of the observations.

(Refer Slide Time: 34:51)

© CET
I.I.T. KGP

$$\vec{d}(n) = [d(1) \ d(2) \ \dots \ d(n)]^T$$

Differentiating eqn.(1) w.r.t. $\vec{w}(n)$

$$\nabla e(n) = -\vec{x}(n)$$
$$\underline{\underline{\vec{J}(n) = -\vec{x}(n)}}$$

So, here d_n is simply defined as d_1, d_2 vector containing d_1, d_2 up to d_n this whole thing transpose. Now, this equation that we had obtain that is e_n equals this, let us call it as equation number one for today. Now, if we differentiate this expression with, this equation with respect to W_n , then what we are getting, in that case we are going to differentiate this with respect W_n . And differentiating this with respect to W_n gives you what the gradient of the error vector, the gradient of the error vector is what we are going to get.

And what will be the differentiation of this term 0, because d_n is independent of W and what will be the differentiation of this expression minus x transpose n , very good. This is going to be minus x transpose n , because this is the expression of earlier for, it is of the form X transpose W already. So, X transposes transpose is going to be, so X the transpose itself will be this term, so that whenever you are ultimately deriving the solution, then it will be transpose of this expression.

So, differentiating, so what we are doing that differentiating equation 1 with respect to the w vectors is going to be grad of e_n , which will be equal to minus X transpose n . And as we already know that the Jacobian matrix is going to be the transpose of the gradient matrix, that we have derived in the last class. Because, gradient matrix is what, gradient matrix is simply going to be the n by m matrix, whereas the Jacobian is going to be m by n .

So, it is just the transpose of that, so that is why the Jacobian matrix J_n is going to be what in terms of X minus X of n , because we have to take the transpose of the gradient matrix. So, transpose of the gradient matrix means, transpose of this once again means we are back again to X_n . So, J_n equal to X_n and what we are simply going to do, this is the very interesting expression that we have got and now we are in a position to this relation how did we obtain, we obtain it from the perceptron equation.

Simple perceptron equation we had obtained that the errors term is equal to X into W X being the input, so this X into W is the perceptron expression. So, we had defined the error that way d minus x times W and this is what we had obtain by simple differentiation that. Now, the Jacobian is already obtain and because, we obtain Jacobian now we can substitute this Jacobian expression, in terms of this input vector into the error expression that we have got.

So, we should substitute this in this weight updating equation, so let me just find out the place where I had derived this. So, in this equation which we had earlier obtained as Gauss-Newton method in the pure form. Here instead of J we can write minus of X agreed, so if we write minus of X in terms in place of J than what we are going to get as the expression for W n plus 1 is as follows. I am Going to write it down and you please take time to verify that.

(Refer Slide Time: 39:29)

© CET
I.I.T. KGP

$$\begin{aligned} \vec{w}(n+1) &= \vec{w}(n) + (\vec{x}^T(n) \vec{x}(n))^{-1} \vec{x}^T(n) (\vec{d}(n) - \vec{x}(n) \vec{w}(n)) \\ &= \vec{w}(n) + (\vec{x}^T(n) \vec{x}(n))^{-1} \vec{x}^T(n) \vec{d}(n) \\ &\quad - (\vec{x}^T(n) \vec{x}(n))^{-1} \vec{x}^T(n) \vec{x}(n) \vec{w}(n) \\ \vec{w}(n+1) &= (\vec{x}^T(n) \vec{x}(n))^{-1} \vec{x}^T(n) \vec{d}(n) \end{aligned}$$

Pseudo-inverse of $\vec{x}(n) : \vec{x}^T(n)$

W n plus 1 is equal to W n and then, what we are going to do is that we are going to do two things, you see we have got an errors expression here that is e n and e n mind you is d n minus X n W n. So, we are going to substitute this equation in place of en over here and in place of J n we are going to substitute minus X of n. So, if we do that then we are going to get W n plus 1 is equal to W n already is there. And then, we are going to get, because already a minus is their the Jacobian term is going to be a minus of X of n.

Now, here there is already a Jacobian term, so which will be equal to minus of X transpose n, so we can write down this, so this minus and minus makes it a plus. So, it will be W n plus, now this term we had J transpose J, which is going to be X X transpose, which will be X X transpose inverse of that. So, this will be X X transpose inverse of that, X transpose n multiplied by no this, this will be X transpose X X transpose X inverse X transpose n and here instead of n I am writing d n minus X n W n.

So, this is what we are getting, now something that you can see, so this equal to W_n plus what is it that we are getting, we are simply getting $X^T X^{-1} X$ transpose $n \times n$ minus what are we going to get. We are going to get this expression $X^T X^{-1} X$, now $X^T X^{-1} X$ W_n , so this inverse and this direct.

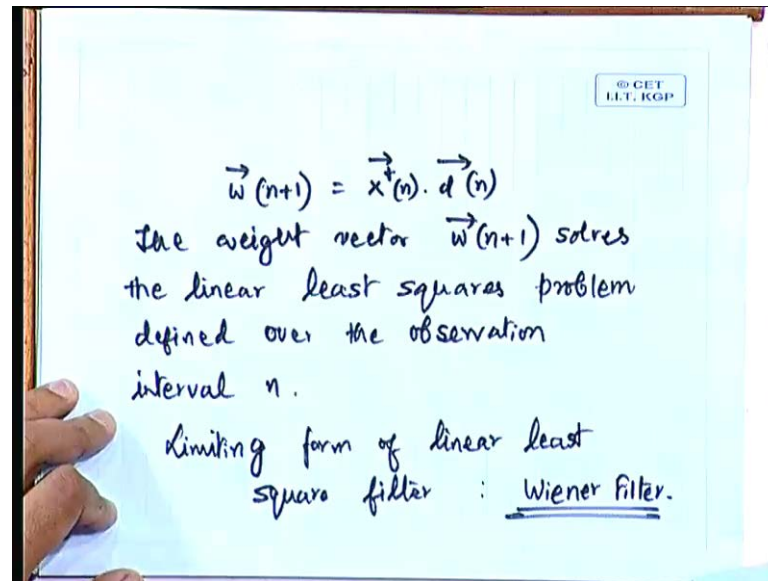
So, what is it going to be identity and then, this multiplied by W_n mean this terms becomes equal to W_n , so what is happening this is plus W_n and this is minus of W_n . So, that is why this term and this term are getting canceled with each other, so then we are getting the net expression as $X^T X^{-1} X$ transpose $n \times n$ as the new expression for W_{n+1} .

Now, this is very interesting, you see that earlier we had the W_{n+1} expression written in terms of W_n , that means to say that W_{n+1} , we would have obtained alternatively means we needed the previous values of W in order to compute that. So, it was a recursive form of relationship that was established earlier, whereas this expression can be computed using single iteration, because all that we require is the n th input data. And if the n th input data is available to us we can compute what the $n+1$ th weight is going to be.

So, W_{n+1} in this case is expressible directly in terms of this, now this expression that is $X^T X^{-1} X$ transpose this whole inverse times X transpose, this term is recognized as the pseudo inverse of the data matrix X_n . This is the pseudo inverse this is recognized as the pseudo inverse of X_n , in fact there is a proof for that which sum of the researches had already shown.

That this is recognizable as the pseudo inverse of X_n and we are going to indicate that as X_n^+ , this being the pseudo inverse expression. So, that is why here we are going to get the solution as W_{n+1} equal to X_n^+ , which is the pseudo inverse of X_n , so we are going to obtain, W_{n+1} as the pseudo inverse of X_n .

(Refer Slide Time: 45:34)

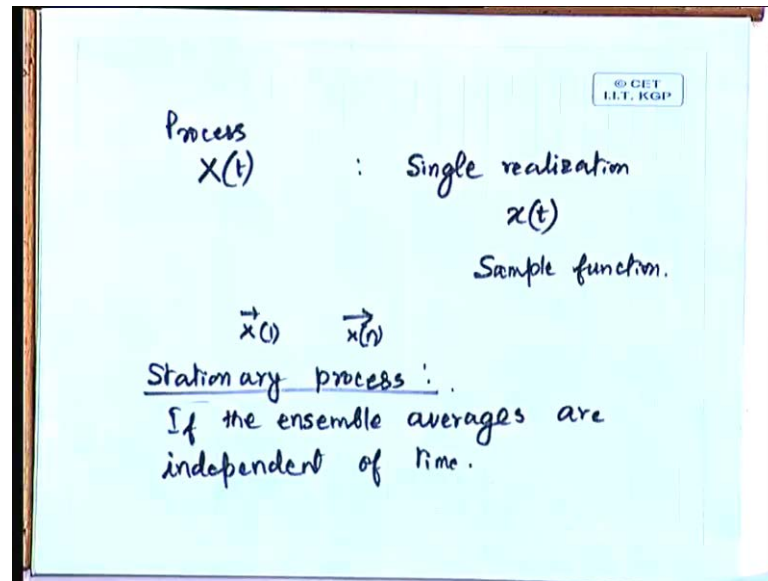


So, the weight vector, so thus the weight vector solves the linear least square problem, problem defined over the observation interval n . So, now what we are going to do is that, so this is form of expression that we have obtain and we are going to take a limiting form of the linear least square filter. So, we are going to take a limiting form of this limiting form of linear least square filter and that is called as the Wiener filter.

In fact, we are going to obtain this limiting form as n tends to infinity, but in this case the assumption that we have to make is that, if you are taking this X and d to be a random process, because X is the input, d is the desired response. So, X and d if you take it to be a random process then, we have to assume that this is following an ergodicity property. So, I am not sure that, if you as well as the distant viewer are having some background related to the stationary and ergodic process.

So, that is why we can spend just few minutes to explain that what the stationary process and ergodic processes mean. So, we will be briefly cover that and then, we will go over to the Wiener filter design aspect.

(Refer Slide Time: 48:30)



Now, let us take a process X of t and then, we take any single realization of this process, single realization we are calling as X of t . So, then what happens is that this X of t is nothing but, a sample function of the process X of t . So, this is the process and a single realization of that, we are going to call these thing as a sample function. It is as if to say that you have got one random variable X , supposing in the form for vector you have got a random variable X_1, X_2 up to X_n those are it is a elements, but you have got a random variable.

And you take a single you collect the samples of those variables, let say that X vector 1, X_1 to X_n this you take. In that case this X_1 vector X_2 vector up to X_n vector what you have take is actually a single realization, a single instance of the whole process X . You cannot capture everything, you have only made n number of observations and you have only collected the n number of points out of it. So, this X_1 to X_n the function that it covers can as basically described as a sample function.

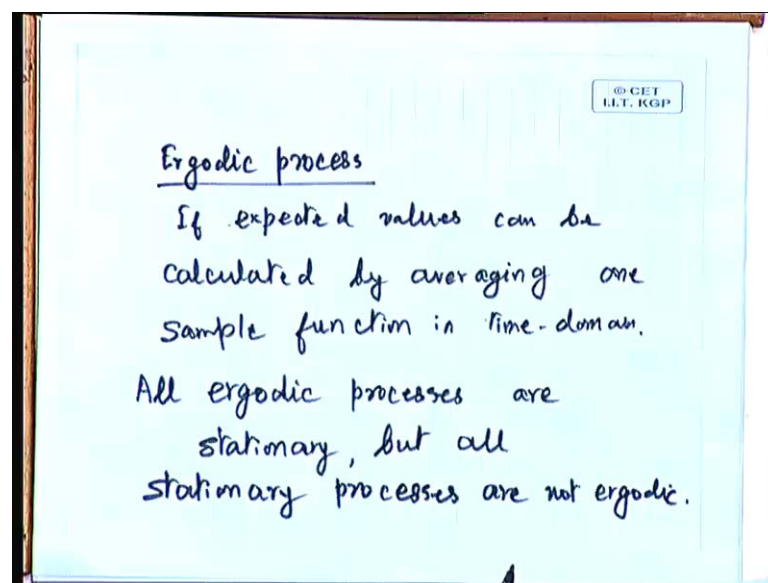
Now, we defined two quantities first is that what is a stationary process. The definition of stationary process is that, if the ensemble averages are independent of time. Just to understand a thing in a very qualitative sense, that stationary process is something where the statistical characteristics do not suddenly deviate. You take the ensemble average now and you take the average at some later instance, then that is not going to varying. So, drastically there is no change in the statistical characteristics.

So, that is being defined as a stationary process, so in fact the stationarity assumption always we have to make in realization of this. Because, what the inherent assumption that you are making is that, when you are collecting the training samples when you are collecting $x_i d_i$'s. You are deriving this $x_i d_i$'s out of a stationary process and you are also going to use it in a stationary environment.

If you are not going to use in an stationary environment, then you have to all the time train it, you have to make it adaptive. So, this is the stationary process definition that if the ensemble averages are independent of time. And ergodic process means, that if the expected values can be calculated by taking by averaging only one sample function that means, to say that here the n observations that we have made.

That is one sample function and if you can calculate the average, if you can calculate the expectation based on this one sample function observation only, then that will be described as an ergodic process.

(Refer Slide Time: 52:39)

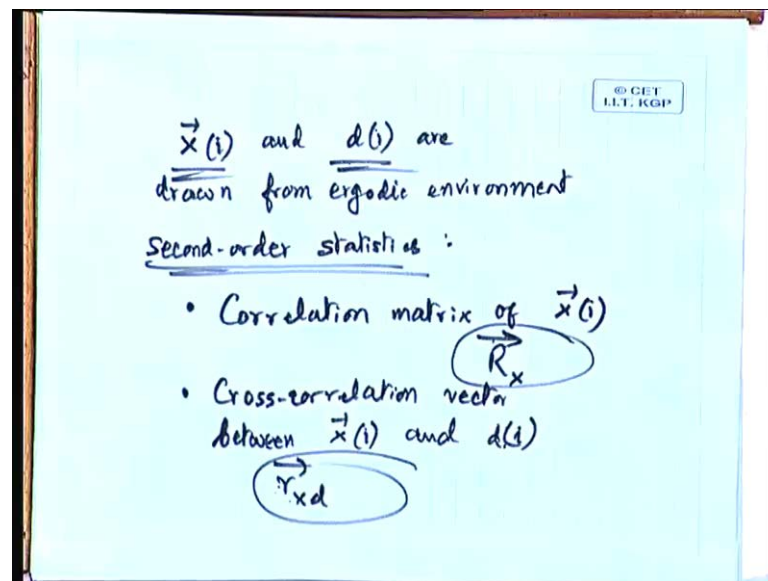


So, the definition of ergodic process says that if expected values can be calculated by averaging one sample function in time domain. So, now can you tell me is ergodic process a stationary process yes or no.

Student: ((Refer Time: 53:28))

Definitely yes, all ergodic functions are going to be stationary, because you are in this case saying that using one sample function time domain itself you can calculate the expected values. So, all ergodic processes are stationary, but all stationary processes are not ergodic. So, what we are making as the assumption is that we are drawing this X vector and the d from ergodic environment.

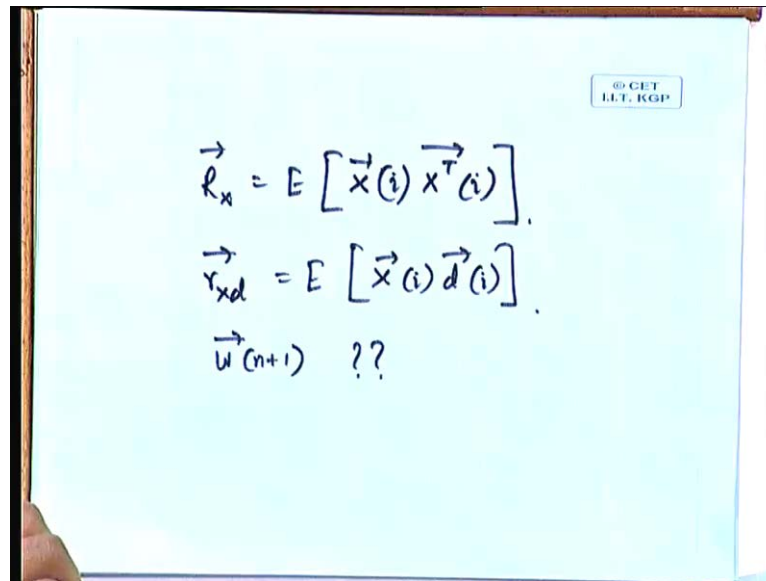
(Refer Slide Time: 54:38)



So, our assumption is that X_i and d_i are drawn from ergodic environment, that means to say that whatever expectations we get that should be derivable from this sample function itself X_i d_i 's sample function. Now, this kind of environment is actually described by second order statistics. So, we define some second order statistics for this ergodic environment and those second order statistics are firstly the correlation matrix of X_i . In fact, correlation matrix will be defined as, we will be indicating by the matrix R_x .

And the other second order statistics that we make use of is a cross correlation between this X_i and d_i , so this will be the cross correlation vector between X_i and d_i and this will be denoted by r_{xd} . So, these two quantities they defined the second order statistics. And we are going to define these terms R_x , in fact you must be knowing that in a very simple terms, where we will expand this in the next class, but talking in their basic definitions.

(Refer Slide Time: 56:37)


$$\vec{R}_x = E [\vec{x}(i) \vec{x}^T(i)]$$
$$\vec{r}_{xd} = E [\vec{x}(i) \vec{d}(i)]$$
$$\vec{w}^{(n+1)} \quad ??$$

This R_x is going to be the expectation of $X X$ transpose, so $X X$ transpose is what, $X X$ transpose is going to be the outer product of these two vectors. So, that is going to be an n by m matrix, so this is the expectation of $X_i X_i$ transpose, this is the correlation matrix and r_{xd} 's definition they cross correlation vector definition is that, it is expectation of $X_i d_i$. So, in terms of these two quantities we will try to express the updated weight.

So, can we get this updated weight, in terms of the correlation matrix and the cross correlation vector between X_i and d_i . That is something that we are going to see in the coming lecture.

Thank you very much.