

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

Lecture -15
Least Mean Squares Algorithm

Squares Algorithm and as before, we start the actual topic for today, we have firstly some unfinished parts pertaining to the last lecture. And I would again like to begin with some doubts which were expressed at the end of the class last time. Now, last time we were actually going through the linear least square filter.

(Refer Slide Time: 01:47)

$$\vec{w}(n+1) = \vec{w}(n) + \dots$$

$$\vec{w}(n+1) = (\vec{X}^T(n) \vec{X}(n))^{-1} \vec{X}^T(n) \vec{d}(n)$$

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

$\vec{x}(1) \dots \dots n$ -dimensional.

$n \times m$ -dimensional.

And in that we had actually developed 1 quite interesting relation, which is that $\vec{w}(n+1)$ is equal to $(\vec{X}^T(n) \vec{X}(n))^{-1} \vec{X}^T(n) \vec{d}(n)$. This is the relationship that we had got as the $n+1$ th weight of the system. Now, as we discussed last time that, this is actually independent of $\vec{w}(n)$, so it is not a recursive form of the expression. That we are using \vec{w} , without using $\vec{w}(n)$ we are able to use $\vec{w}(n+1)$ as if to say.

That if $\vec{X}(n)$ and $\vec{d}(n)$ they are known to us, then we will be able to compute $\vec{w}(n+1)$, but this had raised one serious doubt at the end of the class last time. That whether we can forget about all the past observations, is it so that it is only the present observation that we are considering. Now, about $\vec{d}(n)$ it is ok, because $\vec{d}(n)$ is the expected output of the

target output of the n th observation, but this one, please note that these are capital X . So, some people must have wrongly thought by assuming this to be small x .

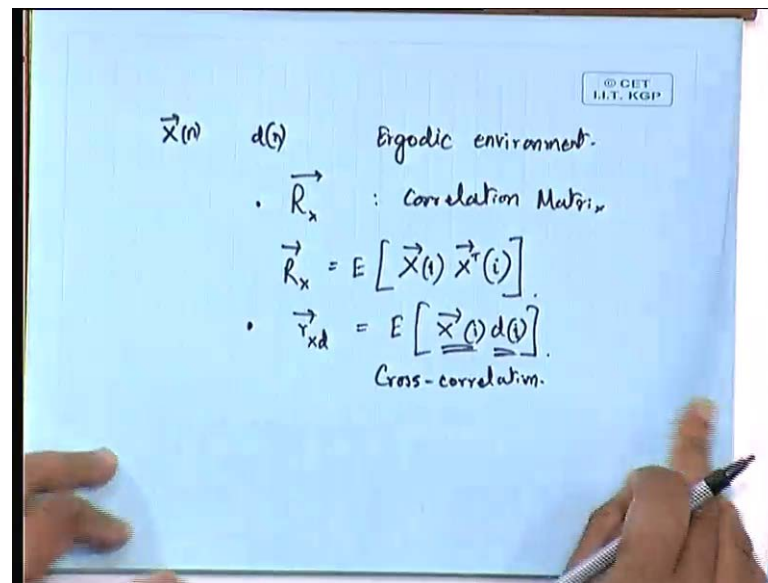
And I had intentionally made one distinction between the small x vector and the capital X vector. Actually if you remember that the capital X vector, we were indicating the capital X vector for n observations as a collection of n such small vector, we should we should write it as X_n should be equal to the vector that we will get out of $X_1 X_2$, so on up to X_n . Where X_1 is nothing but, an m dimensional vector, so this is nothing but, an m dimensional vector which is the first observation.

And like wise x_2 will be the second observation, x_n will be the n th observation and this X_n , this capital X_n matrix that we are getting effectively. In fact, we are getting this we has define this to be this transpose, that means to say that we are going to get it as n by m dimensional matrix. So, those who had wrongly inferred that W_{n+1} is only dependent upon X_n is wrong, because what we mean is that here the entire X_1 to X_n is to be considered.

So, in other words, the matrix X_n includes X_1 to X_n all these observations and we are making a note of all these observations together in this matrix X_n . So, if all these things are known, then only we are able to compute W_{n+1} , so this is the idea that I wanted to convey that in this case. This W_{n+1} please remember that it is indeed dependent upon all the observation.

Only thing is that, in the earlier case you remember that we were deriving W_{n+1} in terms of W_n plus something, which was in fact, a recursive relationship were as here this convert just to a single iteration. Because, all that we need to know is the set of X_n 's, the set of inputs of n observations and the n th target output. So, this is one of the things that we were discussing and then, towards the end of the class actually we were considering the limiting case of the linear least square filter.

(Refer Slide Time: 06:45)



And as we said that in order to describe the limiting form of the linear least square filter we had assumed that, this X_n and d_n they are basically taken from an ergodic process, they are basically taken from an ergodic environment. So, as I discussed in the last class that ergodic necessarily means that it is a stationery environment. So, that we will be able to obtain the expectation by averaging only one sample function in the time domain.

So, from this consideration we said that such an environment will be described by in terms of second order statistics, where we were considering we had define two things. One is the correlation matrix of x , which we are calling as the R_x which is nothing but, the correlation matrix, which we had defined last time as R_x being equal to the expectation of $X_i X_i^T$ matrix to say X^T transpose i . So, this is the correlation matrix, the definition of that.

And the second parameter that we had defined is the cross correlation vector, between X_i and d_i . So, that we were defining as R_{xd} which was. In fact, define as the expectation of $X_i d_i$, so this is the cross correlation. And this is the cross correlation between this X_i and the d_i . So, now let us see that how we can express this R_x and R_{xd} .

(Refer Slide Time: 08:54)

$$\begin{aligned}\vec{R}_x &= E[\vec{x}(i) \vec{x}^T(i)] \\ &= \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \vec{x}(i) \vec{x}^T(i) \\ &= \lim_{n \rightarrow \infty} \frac{1}{n} \vec{X}(n) \vec{X}^T(n) \\ \vec{r}_{xd} &= E[\vec{x}(i) d(i)] \\ &= \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \vec{x}(i) d(i) \\ &= \lim_{n \rightarrow \infty} \frac{1}{n} \vec{X}(n) d(n)\end{aligned}$$

In fact, the correlation matrix R_x can now be expressed as follows, that R_x is the expectation of $x_i x_i^T$, which is equal to the limit of n tending to infinity, this is because it is ergodic. So, we are taking n number of observations where n is sufficiently large, so that is why we are having n tending to infinity and what it means is that we have to sum up these.

So, what are these we have to sum it up? Sum of this $X_i X_i^T$, this we have to sum up over i is equal to 1 to n and then, we have to divide it by 1 upon n , in order to get this R_x . So, in fact if we can of course, take it to be the, by applying this definition it is possible for us to say this is again the small x 's. Please remember these are the small x 's, so this is x_i , so x_i means the m dimensional input vector, x_i is the i th observation of the input vector x transports i is just the transpose of that x_i .

So, this is the i th observation and we have got n such observation, so this in other words can be expressed as the limit intending to infinity. And we can express it as 1 by n capital X transpose n , times X_n , so this is what we are getting as the R_x . So, this is the limiting form of the equation of R_x and the cross correlation vector that is R_{xd} , that can be expressed as the expectation of $X_i d_i$, which is equal to the limit n tending to infinity 1 by n summation i is equal to 1 to n $X_i d_i$.

Again X_i here is the small x_i small $x_i d_i$, which becomes equal to the limit n tending to infinity 1 by n capital X transpose n d_n , so these two quantities are expressed as follows

this the correlation matrix R_x . And the cross correlation between the x and d is and then, we can formulate the linear least square solution as follows.

(Refer Slide Time: 12:26)

Wiener solution

$$\begin{aligned}\vec{w}_0 &= \lim_{n \rightarrow \infty} \vec{w}(n+1) \\ &= \lim_{n \rightarrow \infty} \left(\vec{x}^T(n) \vec{x}(n) \right)^{-1} \vec{x}^T(n) \vec{d}(n) \\ &= \lim_{n \rightarrow \infty} \frac{1}{n} \left(\frac{1}{n} \vec{x}^T(n) \vec{x}(n) \right)^{-1} \lim_{n \rightarrow \infty} \frac{1}{n} \vec{x}^T(n) \vec{d}(n) \\ &= \vec{R}_x^{-1} \vec{r}_{xd} \\ \vec{R}_x &= \lim_{n \rightarrow \infty} \left[\frac{1}{n} \vec{x}^T(n) \vec{x}(n) \right]\end{aligned}$$

You see the linear least square solution as we had obtained is for $\vec{w}(n+1)$, we are obtaining the $\vec{w}(n+1)$ th vector, which can be defined as the limiting vector \vec{w} . Where this $\vec{w}(n+1)$ we have to compute as the limit n tends to infinity, so the limiting solution of these least square filter is limit of $\vec{w}(n+1)$ as n tends to infinity. And this limiting solution is also called as the Wiener solution, so the expression that we are getting for \vec{w} is the Wiener solution.

And now, here we can substitute the expression for $\vec{w}(n+1)$ that we had already obtained for the linear least square filter. What is the relation that we had got, we had we had got these expression the $\vec{w}(n+1)$ was equal to $(\vec{x}^T(n) \vec{x}(n))^{-1} \vec{x}^T(n) \vec{d}(n)$. This itself was defined to be the pseudo inverse of $\vec{x}(n) \vec{x}^T(n)$ times $\vec{d}(n)$, this is what we have got as $\vec{w}(n+1)$.

So, here in place of $\vec{w}(n+1)$, we can as well substitute that which leads to limit n tending to infinity $(\vec{x}^T(n) \vec{x}(n))^{-1} \vec{x}^T(n) \vec{d}(n)$. Where it is possible to show that this can be split up into product of two limits. And what are they this can be split up into limit n tending to infinity $\frac{1}{n} \vec{x}^T(n) \vec{x}(n)$ inverse times limit n tending to infinity $\frac{1}{n} \vec{x}^T(n) \vec{d}(n)$.

And this by our basic definition is the $R \times$ matrix, this is nothing but, the $R \times$ matrix, so here it becomes $R \times$ inverse, this becomes the $R \times$ inverse matrix times $R \times d$. So, that means, to say that here the Wiener solution that we are getting is a product of these two, yes any questions on that.

Student: ((Refer Time: 15:39))

1 by n

Student: ((Refer Time: 15:46))

Yes, this can be actually shown that if we break it up, if we break up this quantity, then there is a 1 by n square, because you see that here we are taking two products separately like this. So, if we break it up, then it can be shown that it is 1 by n times this.

Student: ((Refer Time: 16:23))

Inside the bracket

Student: ((Refer Time: 16:28))

No, here, no you see this is, no this is the basic definition of $R \times$ you see, so we are applying the basic definition of $R \times$ as follows, so this whole quantity. So, $R \times$ is limit n tending to infinity of this entire quantity.

Student: ((Refer Time: 17:12))

I have got your point, you are saying that $R \times$ inverse yes, $R \times$ inverse is equal to limit n tending to infinity I apply the basic definition, that is 1 by n into x transpose $n \times n$ the whole thing.

Student: ((Refer Time: 17:44))

Whole thing inverse, so this is actually the whole thing inverse, but the question is that becomes $R \times$ inverse, so here your observation is correct that it should be 1 by n should go inside the bracket, very, very correct yes.

Student: ((Refer Time: 18:14))

Yes very very correct, there is no 1 by n square term, there is no 1 by n square term, in fact what happens is that this 1 by n goes into the inverse and that effectively nullifies this in turn. So, this can be shown, we are not going in to the details of the matrix manipulation, but this effectively becomes the $r \times d$ and this becomes the $R \times$ inverse, does it solve the doubt, perfectly thank you very much. So, now what we are, there is some after thoughts that we should get on this.

(Refer Slide Time: 19:11)

© CET
I.I.T, KGP

Wiener: $\vec{w}_0 = \underline{\underline{R_x^{-1}}} \underline{\underline{r_{xd}}}$

Adaptive Filtering

LMS Algorithm.

That here you see that \vec{w}_0 vector is equal to R_x inverse times r_{xd} vector, now here you see that, that means say that solution of this Wiener. So, in order to obtain the Wiener solution, we must have a idea about this R_x inverse and r_{xd} . But, the question is that, if we are knowing the statistical parameters or the statistical nature of the data before hand, then that we are in fact, really going to have.

That beforehand we will be knowing the correlation matrix, as well as the cross correlation between the inputs and the observed outputs. So, since we are not knowing that beforehand, it is not possible for us to calculate this \vec{w}_0 vector in such kind of environments. So, in fact for any practical purpose that means, to say for any unknown environment we have to go in for an adaptive filtering.

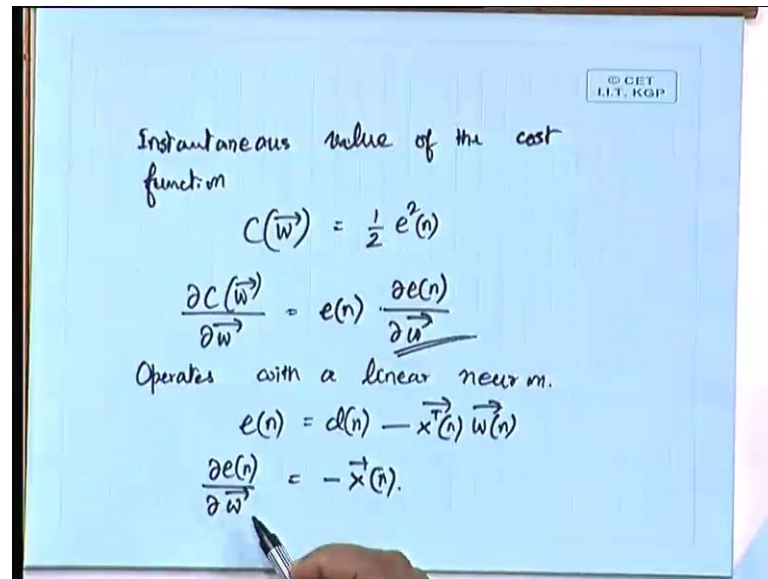
So, we have to go in for what is called as adaptive filtering, and one such kind of adaptive filtering that we are going to adopt is the least mean square or in short form we are going to call it as the LMS. So, we are going to describe the LMS algorithm now. Now, LMS algorithm mind you has got very much similarity to the linear least square, that we have formulated, but the major dissimilarity lies in the fact.

That in the case of linear least squares, we were taking the cost function to be the summation of all the error squares. So, the cost function C_W that we were taking was actually e_1^2 plus e_2^2 square up to e_n^2 square. Whereas in the case of least mean square we are taking only, as the cost function we are taking only the instantaneous value

of the error. That means, to say the cost function C W calculated at w is equal to W n is simply equal to e n square.

Or rather half of e n square, because we are all the time inserting that factor of half, because of our convenience in computing the derivative.

(Refer Slide Time: 21:56)



Instantaneous value of the cost function

$$C(\vec{w}) = \frac{1}{2} e^2(n)$$

$$\frac{\partial C(\vec{w})}{\partial \vec{w}} = e(n) \frac{\partial e(n)}{\partial \vec{w}}$$

Operates with a linear neuron.

$$e(n) = d(n) - \vec{x}^T(n) \vec{w}(n)$$

$$\frac{\partial e(n)}{\partial \vec{w}} = -\vec{x}(n).$$

So, LMS takes the instantaneous value of the cost function, which is C W is equal to half of e n square or e square n . Now, differentiating, now if we differentiate this C W with respect to the W vector what we get is we get e n times $\text{d} e(n) / \text{d} \vec{w}$. Now, like the linearly square filter, again here we are operating with a linear neuron, so this operates with a linear neuron.

So, that it is possible for us to calculate this $\text{d} e(n) / \text{d} \vec{w}$ easily, because we know that for a linear neuron e n is equal to e n is equal to d n minus X transpose W X transpose n W n . So, that is why we can compute $\text{d} e(n) / \text{d} \vec{w}$ is equal to, what is equal to minus X n simply, because this is X transpose. So, differentiating by the vector means it is the transpose of the transpose and this becomes minus X n .

We are differentiating with respect to W over here and the derivative of d n with reference to W is definitely going to be 0. So, this is equal to minus X of n , so that it is possible for us to write that $\text{d} C / \text{d} \vec{w}$.

(Refer Slide Time: 24:16)

$$\frac{\partial C(\vec{w})}{\partial \vec{w}} = -\underline{e(n)} \vec{x(n)}$$

"stochastic" steepest descent.
 $\vec{w}(0)$

$$\hat{g}(n) = -\vec{x(n)} e(n)$$

$$\vec{w}(n+1) = \vec{w}(n) + \eta \vec{x(n)} e(n)$$

So, we can write $\frac{\partial C}{\partial \vec{w}}$ as equal to minus $\vec{x}^T e$ and this is actually the estimate of the gradient mind you that this is not the exact gradient. Because, for exact gradient we would have required the error as a vector e_1, e_2, \dots, e_n we do have got, but in this case we have got only the e_n . So, we can express these the derivative of C with reference to \vec{w} as the gradient estimate.

So, gradient estimate can now be written as in fact, whether we write it as minus $\vec{x}^T e$ or whether we write it as, because in this case e_n is a scalar quantity, e_n is equal to d_n minus this. So, we are taking only one neuron and this is the n th observation of that, so e_n is a scalar d_n is a scalar. And this $\vec{x}^T \vec{w}$, that itself becomes a scalar although individually \vec{x} is a vector and \vec{w} is a vector, so this is simply the linear neuron model, that is what we are considering.

So, this minus $\vec{x}^T e$ can also be written as $\vec{x}^T e$ does not matter, so this is the estimate of the gradient. So, since this is estimate we are indicating as \hat{g} not as g exactly. In the case of steepest gradient we were indicating it has g_n , but here we are writing it as the estimated \hat{g}_n . And because it is equal to minus $\vec{x}^T e$ of this quantity, what is going to be our updating equation or what is the LMS algorithm equation.

In fact, are we correct in writing that $\vec{w}(n+1)$ is equal to $\vec{w}(n)$ plus, because it is the gradient, so we have to move in a direction opposite to the gradient. So, that is why it is

$W_n + \eta X_n e_n$, where $X_n e_n$ or ηg_n , or ηg_n is equal to minus, g_n is equal to minus $X_n e_n$. So, it becomes W_{n+1} becomes equal to $W_n + \eta X_n e_n$.

Nobody is objecting to it, is it the same as the that of the gradient decent that we have discussed, steep as decent, steepest in decent approach that we have discussed is it same as that, exactly same as that, it is not. Because, after all we have made an estimate of the gradient and because we have made a estimate of the gradient even the weight that we are computing is also an estimated weight.

So, it is not right for us to indicate this by simply putting W or W vector, we should put it as W cap vector both for $n+1$ and n . So, W_{n+1} is equal to $W_n + \eta x e_n$, this is our weight updating equation for the LMS algorithm. So, can we now call it, because our basic objective was to make it an adaptive filter, now can we call it an adaptive filter yes are no. Anybody disagreeing that it is not an adaptive filter it is an adaptive filter, but why is it an adaptive filter.

You see we are starting with the first observation, so what is our error our error is e_1 , so we are having half of e_1 square to be our cost function, we begin with any arbitrary weight to start with we have W_0 as the initial weight. So, we start with any arbitrary initial weight assignment W_0 and first time we operate with e_1 , which we are getting as the difference between d_1 . D_1 is the expected output for the first pattern, d_1 minus X_1 transpose W_1 or W_0 here.

Or rather to say the first one if we you are calling as zero vector, then everything is your e_0 will be equal to this, so your observations will be actually e_0 onwards. W_0 is the starting weight and you are getting the first error. And then, based on that first error you are updating your weight and the weights are getting updated as you are learning more and more. So that means, to say that even if the environment changes, then also we are updating the weight according to the changes that we are observing in the observations.

It is not that we are taking all the n observation together and then, doing the updating we are doing the updating with every observation. So, there lies one great thing, but one thing is there that here we are not getting the exact idea about the weight, we are getting the estimated weight. If we are comparing it with respect to the steepest decent approach. Now, one thing which we had pointed out for the case of steepest decent approach is that, the steepest decent approach follows a definite trajectory.

As we iteratively go on the steepest decent will follow a definite trajectory, remember that we had illustrated that with a simple two dimensional space where we were taking just two vectors W_1 and W_2 . And we were considering the controls like this and then, we were considering one operating point. And then, it was going into the solution following a fixed trajectory. Here the in that case the weight that we were obtaining, where the exact ways whereas in this case this is going to be estimated way.

So, you cannot expect as good a trajectory that you could imagine in the case of steepest decent, because what we are doing is actually an approximation to the steepest decent. So, that is why the trajectory that we will be getting is more like a random there will be some randomness in the trajectory. So, that is why this approach, which is actually in tune with steepest decent, but not exactly the steepest decent, is very often refer to as the stochastic steepest decent.

So, here the very fact that we have use the word stochastic before the steepest descent, means that it is behavior is having some deviations from the perfect steepest descent, that we are going to have. So, here we will be having such kind of variations. Now, we are going to consider the convergence aspects of this LMS algorithm, but before we going to that let me again invite some questions from the participants here, any questions. So, we go over to the convergence consideration.

(Refer Slide Time: 33:30)

© GET
I.I.T. KGP

Convergence considerations in LMS algorithm.

1. Convergence of the mean

$$E[\hat{W}(n)] \rightarrow W_0 \quad \text{as } n \rightarrow \infty$$
2. $E[e^2(n)] \rightarrow \text{constant as } n \rightarrow \infty$
Practical Consistence.

The graph shows a decaying signal, likely representing the error signal $e(n)$ or its square $e^2(n)$ over time n .

Now, one point that we note here is you say that the equation that we had obtained $I W_n$ plus 1 or $W_{cap} n$ plus 1 is equal to $W_{cap} n$ plus η of this terms. And what is after all η of this term it is dependent on e_n and e_n is nothing but, d_n minus $x_n w_n$. So, in other words the weight that we are getting is following some dependence on the x_n and d_n .

Now, in this case one thing that we should consider is that a of course, and first and for most it very much dependence upon this η . So, that is why any convergence criteria. So, if we fix up this x_n for example, if we fix up this n , that means to say that we fix up the environment then the e_n 's are also fixed up because e_n 's then well depend only upon the weight because if x_n 's are fixed.

Then like wise the d_n 's are also fixed, if you are fixing the patterns then there expected outputs are also fixed. So, that is why the e_n 's that you are getting is only dependent on weight. So, in other words we can say that the weight updating equation will be dependent upon η only, for fixed n , for fixed set of inputs the weight updating equation will very much depend upon η .

And as we discussed before that from all practical and convergence criteria's, we have to take very small values of η . It is a mass to that we have to take very small η 's. Now what are the convergence consideration, that we are going to take. The first criteria that we normally take is the convergence of the mean. Convergence of the mean and this is described as the expectation of w_n , expectation of $W_{cap} n$.

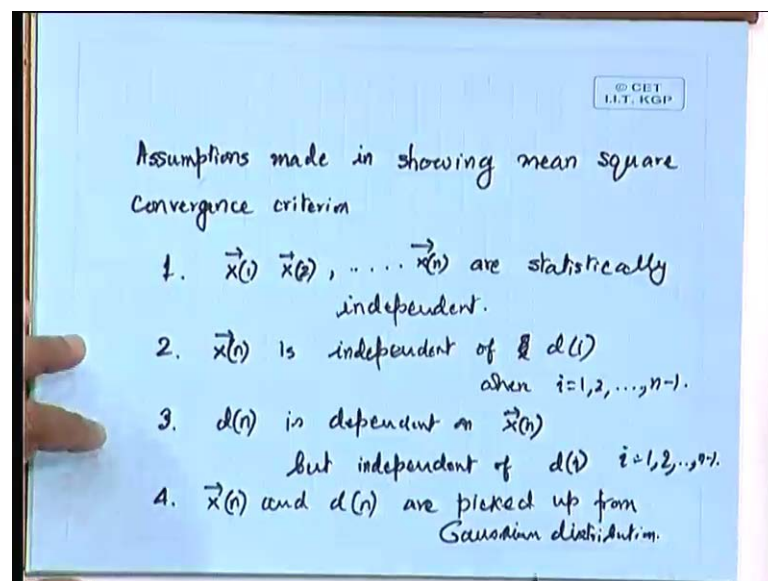
In fact, as n as n tends to infinity if you make the number of observations to be large, then what is it that its it is going to be what can you expect to be the expectation of W_n equal to w_{naught} , yes. You can expect it to be w_{naught} , which is the Wiener solution. So, expectation of w_n is tends to w_{naught} as n tends to infinity, but this is of little practical value. Since a sequence of zero mean, but otherwise arbitrary random vector can also converge in this sense.

And from a practical point of view better convergence to criteria that we normally see is that the expectation of its square error, that means to say e of e square n that becomes constant as n tends to infinity. So, in other words that if it becomes if the expectation of the error that becomes constant at as n tends to infinity.

Then also we can say that the LMS algorithm has converged, yes or no we can say that because, if the error, if there is no change in the error if it has already gone into an constant error with iterations. That means to say that it is characteristic is something like this where this is the error of the cost function, then the cost function will get a fixed value here. So, this thing is a constant is of good measure that it has actually converged.

So, for all practical consideration, we will be taking the convergence of the error, convergence of error to be, convergence of mean square error as the LMS algorithm convergence consideration. In fact, the convergence analysis is very much detailed involves quite complicated mathematics. So, that is why people have solved this problem only under some assumptions and those assumptions mind you are more all less this reasonable assumptions.

(Refer Slide Time: 39:21)



And what are the assumptions that people have adopted to solve the convergence problem. So, assumptions made in showing the mean square convergence criteria is number 1 assumption is that this x_1 the small x_1, x_2 upto x_n they are all statistically independent.

So, all the observations that you are having all the inputs that you are having they are statistically independent inputs. The second assumption is that the input x_n the n th input x_n is independent here all this x 's are the small x please do not feel confused. So, this x

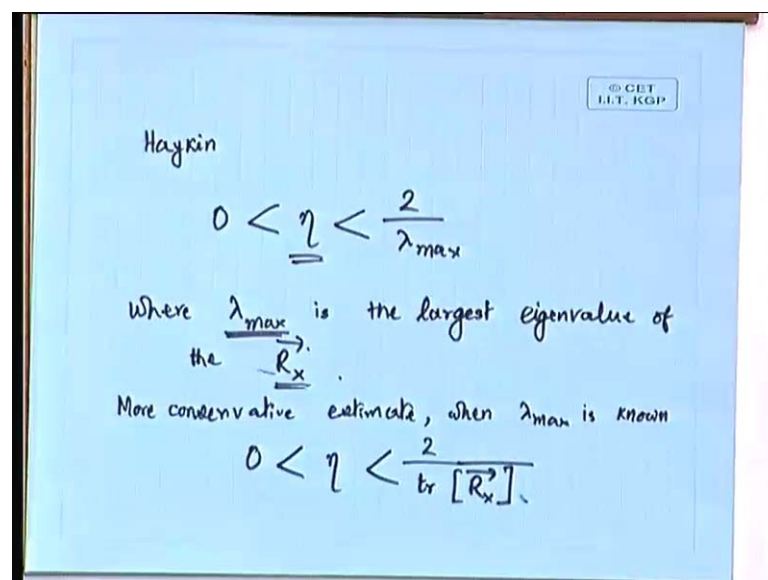
x_n the n th x , that means to say is independent of d_i , where i is equal to 1 2 up to n minus 1.

That means to say that the n th input is independent of the previous target outputs not unreasonable to assume that. The third consideration is that although d_n is dependent on x_n very much. d_n has to depend upon x_n , but d_n is independent of d_1, d_2 up to d_{n-1} . Independent of d_i , i is equal to 1 to n minus 1. The fourth assumption is that this x_n and d_n they are picked up from a Gaussian distribution, because unless the distribution of this x_n 's are assumed in some form.

It is not possible to show the convergence criteria, so the convergence criteria can be shown only under some distribution. And in absence of any definite knowledge about the exact distribution, it is not unreasonable to assume that x_n and d_n will follow a Gaussian distribution. So, we pick up the x_n and distribute x_n and d_n out of a Gaussian distributed variable.

So, now if we assume these four things, then it can be shown that the LMS algorithm converges in the mean square error sense. If this four are valid, then the LMS algorithm converges in the mean square error sense and this has been actually shown by Haykin.

(Refer Slide Time: 43:14)



And Haykin has shown that under these assumptions under these 4 assumptions. The LMS algorithm is convergent in the mean square sense provided that the value of eta lies

within two and lambda max with is 0 and 2 by lambda max, where lambda max is the largest eigenvalue of the R_x matrix.

So, it is the largest Eigen value of the correlation matrix. Now, in a typical application actually the idea of this lambda max is not available. And in such kind of cases where the lambda max is not available, then a more conservative estimate about this eta more conservative estimate. That one can have is more conservative estimate, when this lambda max, is not known is simply by looking at the trace of the matrix R_x .

If you are taking the trace of that matrix R_x that can be approximated for lambda max. So, a more conservative estimate for eta is where eta lies between 2, 0 and 2 by trace of R_x . And what is the trace of R_x , what is the significance of the trace of R_x ?

Students: ((Refer Time: 45:14))

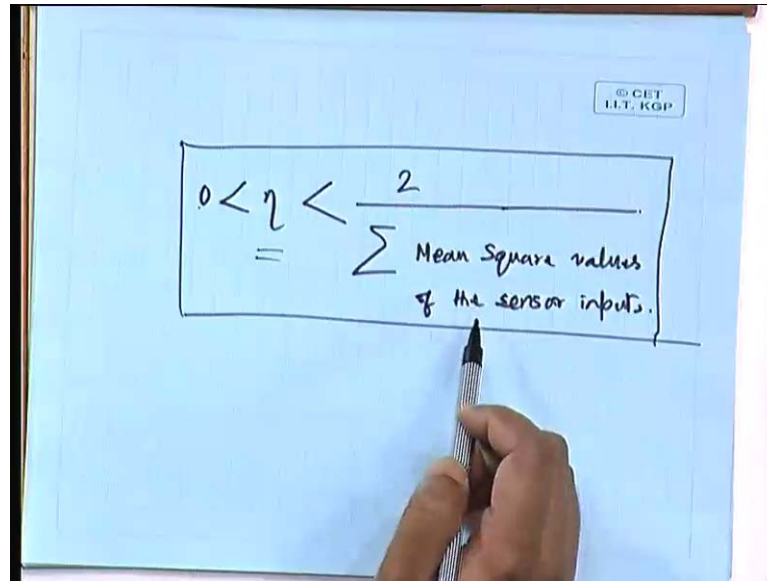
Some of the diagonal elements, yes very good, then the, so that is the some of the diagonal elements and what are the diagonal elements signifying in the case of the correlation matrix.

Students: ((Refer Time: 45:28))

Yes, the power terms, power terms means the means square values of the sensor inputs, because after all we have got the R_x matrix as the expectation of capital X transpose capital X . And what was capital X capital X was nothing, but the matrix that we had formed out of the input vectors x_1 to x_n .

So, that was actually composing the big X matrix and that big X matrix that we had got. Its diagonal elements will be having all the x_1 squared term, x_2 square term, x_3 square term upto x_n square term. So the trace of the matrix in fact, will be indicating the means square values of the sensor input.

(Refer Slide Time: 46:19)



A hand-drawn equation on a blue notepad. The equation is enclosed in a rectangular box and reads: $0 < \eta < \frac{2}{\sum \text{Mean Square values of the sensor inputs.}}$. A hand holding a pen is visible at the bottom right, pointing towards the equation. In the top right corner of the notepad, there is a small logo that says "© CET I.I.T. KGP".

So, what we will be getting is that the value of eta should lie between 0 and 2 by the summation of mean square values of the sensor input. So, you can see that for all practical consideration, the convergence that Haykin has shown under all this assumption. Basically says that it is convergent in the mean square error sense only when the value of eta is within this. Within this are a more conservative estimate would say that within this.

So, this is very easy to compute, because we know that what our inputs are and if you now that about the inputs are, we simply adopt the inputs, thus the square of those inputs. And we sum it up of course, we have to make large number of observations on that and then if we have 2 by that quantity, that will give us the upper limit of eta and if eta is restricted to this bounds, then the convergence of the elements algorithm is guaranteed.

So, this is and in fact, any large values of eta; obviously, will make the system unstable, because this aspect we discussed earlier also that the convergence will be only restricted for the case of the smaller etas. Another consideration that we should have is, you see that, let us go back again to this equation that W_{n+1} is equal to w_n plus eta times x_{n+1} .

Now, what is the effect of small eta and what is the effect of large eta on the learning itself. If you are having a small value of eta then; obviously, one point is there that it learns slowly. But when it learns it learns all the inputs that had been presented, so far

mind you, because we are iterating upon this algorithm for n equal to 1, 2 etcetera, etcetera, right. So, every time we are making contributions which are getting embedded. So, what happens is that when we compute w_{n+1} this quantity.

First time it does this incremental learning and gets in to this updated w_{n+1} , then these w_{n+1} is already containing. This w_{n+1} will be used in the next iteration $n+1$ of w_n . That means to say that it has already learnt the first one.

Now, if η is small then another small incremental learning is added to it, so that in the n this w_n will contain the learning that has been contributed from everything. Whereas if the value of η is quite large, then what happens, then what happens is that as compared to w_n this term will contribute more.

That means to say that current input that you are providing is contributing more to the learning rather than the agglomerated effect of all the past inputs. So, a small η makes a contribution of all the inputs that we have presented. So, far where as a large η means that it gives more baseness towards the input that is presented at the last time.

So, that is why from a memory point of view from a system memory consideration, we should say that a system that is having smaller values of η has got better memory as compared to the systems which are having larger values of η . So, it is not only the convergence criteria, but even the learning also is better. But only thing is that a small η , the price we pay for a small η is nothing, but the slower convergence that is all other we are gainer.

So in fact, coming to the pros and cons of the least mean square approach, we can say the least mean square approach is definitely a very effective linear adaptive filter. There can not be any doubt about it an it. So, simple implement because it is cost function is just formulated as the instantaneous error square term. And the only price that we pay for in the least mean square algorithm is the slow convergence any question.

Thank you.