Introduction To Adaptive Signal Processing

Prof. Mrityunjoy Chakraborty
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture No # 30

Affine Projection Algorithm (APA)

Ok, we considered normalized LMS algorithm. LMS, we discussed is convergence, you know derivation, convergence, approximately convergence that is it acts like LMS algorithm actually with mu time varying mu with a mu which is different now mu by tracer and so, that effective mu will have upper bound for convergence is 2 mu should be greater than 2 less than 2 greater than 0 all these were discussed. I also showed that if the input is highly correlated then that LMS algorithm requires many iterations to converge, but if the input is white then they convert ideally in the absence of noise just in 2 steps, but for general input which has lot of correlation in highly correlated input LMS algorithm suffers from poorer convergence ok. So, a better alternative here is a generalization of the LMS algorithm called Affine Projection APA. Again, here we will assume all the real case no complex data and though all this can be extended to complex case also, but we will not consider here in this course and also to start with we will not assume any noise. So, we remember what we had in this is NLMS if we had just 2 coefficients w0 cap and w1 cap and there is no noise.

So, what was coming out was dN and this is xN ok. So, since we do not know these coefficients, we take general w0 and unknown w0 xN plus w1 xN minus 1 equal to dN ok.

$$w_0x(n) + w_1x(n-1) = d(n)$$

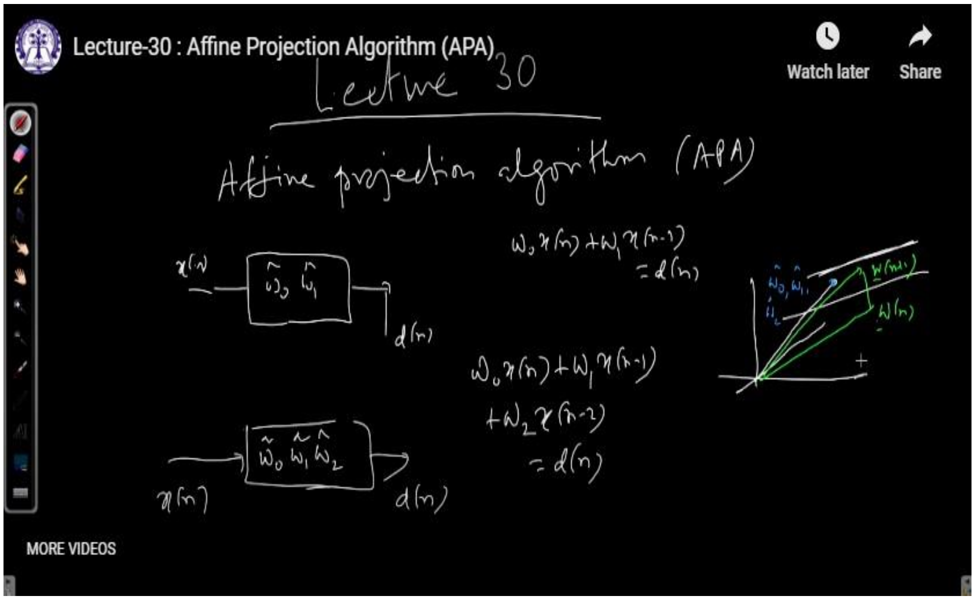
So, there are infinite solutions this actually gives a straight line we have seen ok. At any point on the straight line is a solution of which this is also a valid solution ok, but only thing we do not know.

And of course, you remember the procedure that if wN is the current estimate then from wN we try to arrive at a point on this straight line ok. So, that this equation is satisfied, but distance from that point to wN is minimum that is it is orthogonal perpendicular to this line and so on and so forth. In fact, you can see one thing if we have 3 coefficients suppose what is in this case where I have got 3 coefficients. The same logic will hold here it will be w0 general w0 xN equal to dN. So, this is you know it has 3 x's w0 w1 w2.

$$w_0 x(n) + w_1 x(n-1) + w_2 x(n-2) = d(n)$$

So, 3-dimensional plot and this will be a plane. So, it could be a plane you know like this is a plane and somewhere in the plane I have got this point optimal point this point and then we take a wN then we project on this plane earlier it was straight line now you project on this plane this will take to be wN plus 1. Then we move to N from Nth index to N plus 1th index you get another plane. So, you project this point you know this vector on that plane get wN plus 2 and so on and so forth ok. But at a time at any index, you just are considering only one plane and on that you are projecting the previous weight vector or current weight vector rather we are projecting orthogonally and that is how you do.

In the case of affine projection we generalize like this.

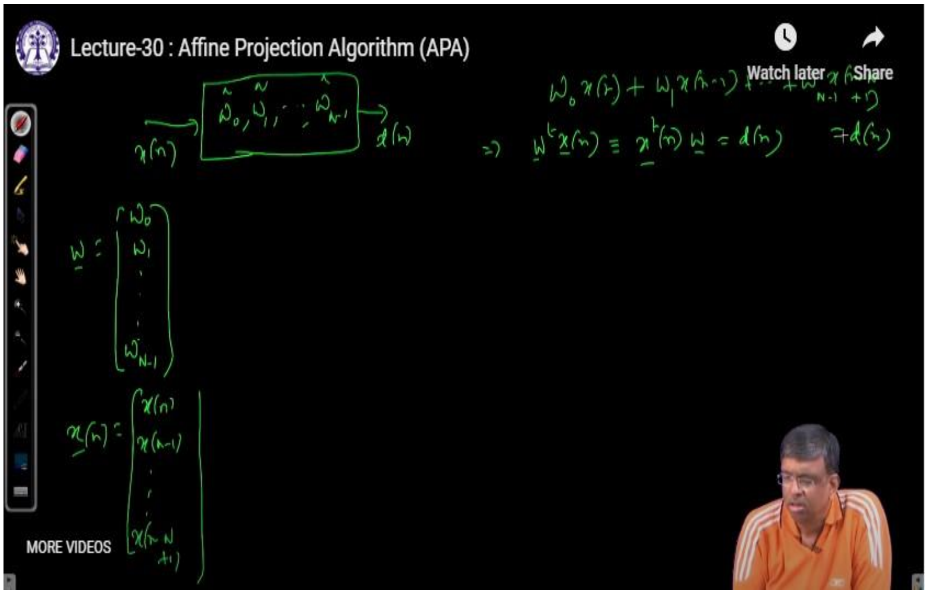


Here just consider a general case these are true parameters general case we have got xN and in the absence of noise we have got dN. Since I do not know these coefficients the general equation, I write it will be in terms of this unknowns w0 xN plus to dN which also you can write as w transpose xN equivalent to x transpose N w is equal to dN w

$$w_0 x(n) + w_1 x(n-1) + \cdots + w_{N-1} x(n-N+1) = d(n)$$

$$\underline{w}^t \underline{x}(n) \equiv \underline{x}^t(n) \underline{w} = d(n)$$

of course, is that coefficient vector unknown coefficients we have to determine the true values and xN is the data vector at the current index N that is current data and some of the past data. So, this is a simple if I have filter output equation this is what you have. Now in the case of APA we try to look find out this w0 to wN minus 1 not just from one plane in fact I should call it hyper plane because it is more than three-dimension plane is in the three dimensions.



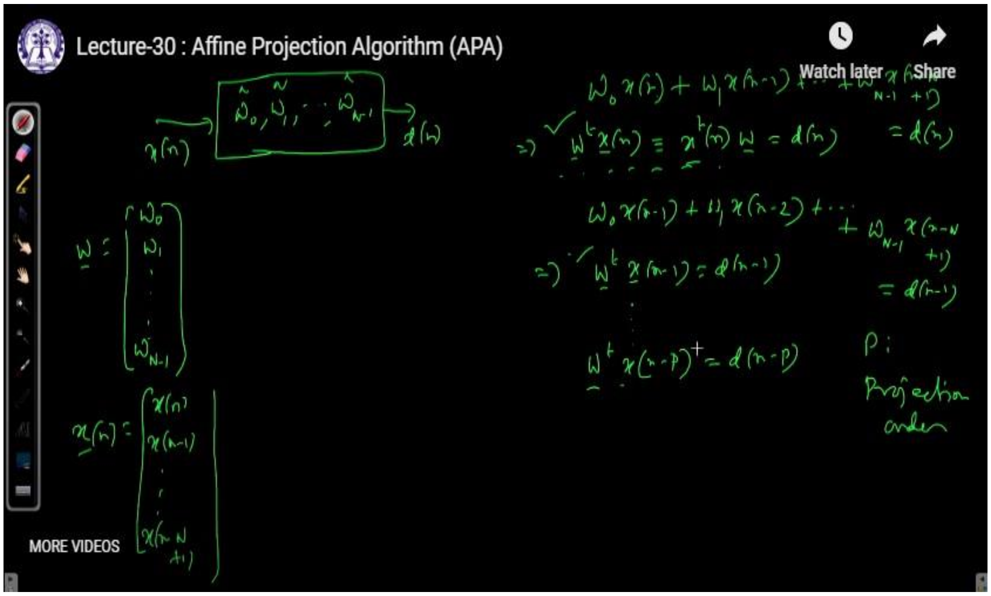
So, let us just use the term hyper plane. So, we try to find out w0 w1 and this wN minus 1 all this not just from one hyper plane. We assume that since the coefficients are not changing at the previous clock N minus 1th clock also if I have this that is at N minus 1th clock if I input xN minus 1, there is a corresponding filter output will be filter coefficients are not changing for the same coefficients at Nth clock these are the data and you get w transpose xN dN at N minus 1th clock it should be dN minus 1, that is w should be such that w transpose xN at Nth clock should be dN the dN that is coming out w transpose at N minus 1th clock it would give you dN minus 1.

$$\underline{w}^t\underline{x}(n-1) = d(n-1)$$

So, simultaneously this should be satisfied this should be satisfied this one this one this one there is an Nth clock w should satisfy this equation same w should satisfy at N minus 1th clock this equation and dot dot dot dot like that up to we go p terms N N minus 1 N minus 2 up to maybe N minus p. p is called projection ordered.

$$\underline{w}^t\underline{x}(n-p) = d(n-p)$$

So, we are I am looking to solve for or obtain a w which simultaneously satisfy not just I mean one hyper plane or one equation one plane or one straight line in the case of two dimension it will satisfy multiple such planar equations hyper plane equations at Nth clock N minus 1th clock N minus 2th clock dot dot dot N minus pth clock simultaneously. That means wherever they intersect my w must belong there because the planes to explain that let me consider just three coefficients only let me consider three coefficients.



Suppose you have one equation which is w0 xN plus w1 xN minus 1 plus w2 xN minus 2 equals to dN that is in short w transpose xN is dN and at N minus 1th clock you have got w transpose xN minus 1 vector there is xN minus 1 N minus 2 N minus 3 is dN minus 1 just 2.

$$w_0 x(n) + w_1 x(n-1) + w_2 x(n-2) = d(n)$$

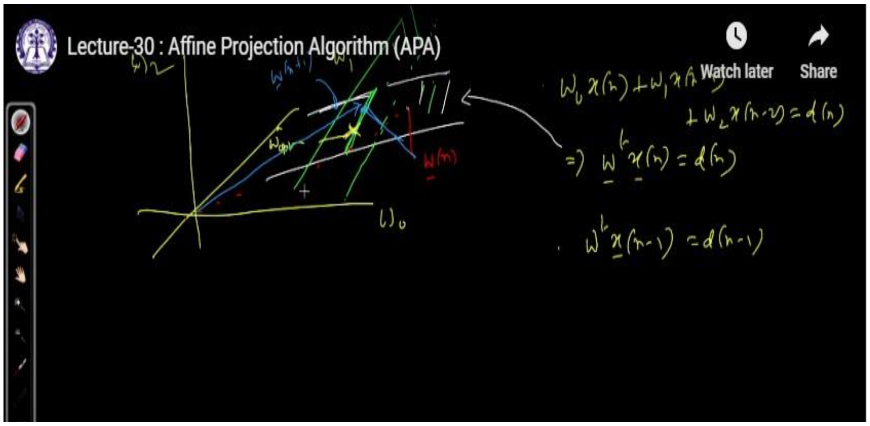
$$\Rightarrow \ \underline{w}^t \underline{x}(n) = d(n)$$

$$\underline{w}^t \underline{x}(n-1) = d(n-1)$$

So, this is one plane. This is one plane that is this is this plane and another one will be another plane another one will be another plane maybe like this. So, these two planes actually will intersect let me try to do a better diagram. Suppose it like this. So, this plane has you know this is this is this plane this is this plane and that is this plane and they intersect the two planes will intersect at one line that you can understand you take two you know piece of papers if they intersect that will be a straight line I mean from three dimension it becomes two dimension that is plane had three variables the intersection will be a straight line we have actually two variables that is obvious we eliminate one of the three w0 or w1 or w2 and you get one equation from this. So, that is this that is obvious. Now, here what will you do is this suppose your wN is somewhere here and optimal one optimal one will be satisfying both these equations these equations.

So, that should lie optimal vector should lie in both the planes therefore, it should lie on the intersection which is common to both the planes. So, maybe this is where your optimal one is w opt the two system parameters. Now, what you are doing if you are doing NLMS from here we will just take a projection on this plane. So, this will be a new vector this will be a new vector. So, this will be wN plus 1, but you see wN plus 1 and w opt there is so much of gap, but in APA affine projection algorithm we will not project wN on this plane or on this plane we will project on that intersection in this case because they are three-dimensional intersection is two dimensional which is straight line.

In general, if each hyper plane is supposing N dimensional then that intersection will be between 2 will be N minus 1 between 3 will be N minus 2 dimensional and like that this you can visualize easily. So, we will take a perpendicular there. So, we will be drawing you know perpendicular on this line which is intersection. So, that will be my wN plus 1 this vector is wN plus 1 under APA and now see this is now much closer to this optimal one, then this guy under NLMS I would have projected here. So, much of distance between the optimal one and the projected point, but if I project on this plane not on this plane or this plane separately, but on the intersection between the two planes then I hit a point which is much closer to the optimal one.



So, I am already very close to optimal one. So, I will require much less iterations to reach the optimal one than the NLMS this is the philosophy this you can extend to the general N dimensional case. That means, optimal the here the optimization will be this we should minimize or before that maybe some definitions will help us at this stage. These equations can be combined into a matrix vector form like this. If I define xN where this is xN xN vector we already defined the input data vector at N s clock then xN minus 1 dot dot dot dot xN minus N plus 1 this xN.

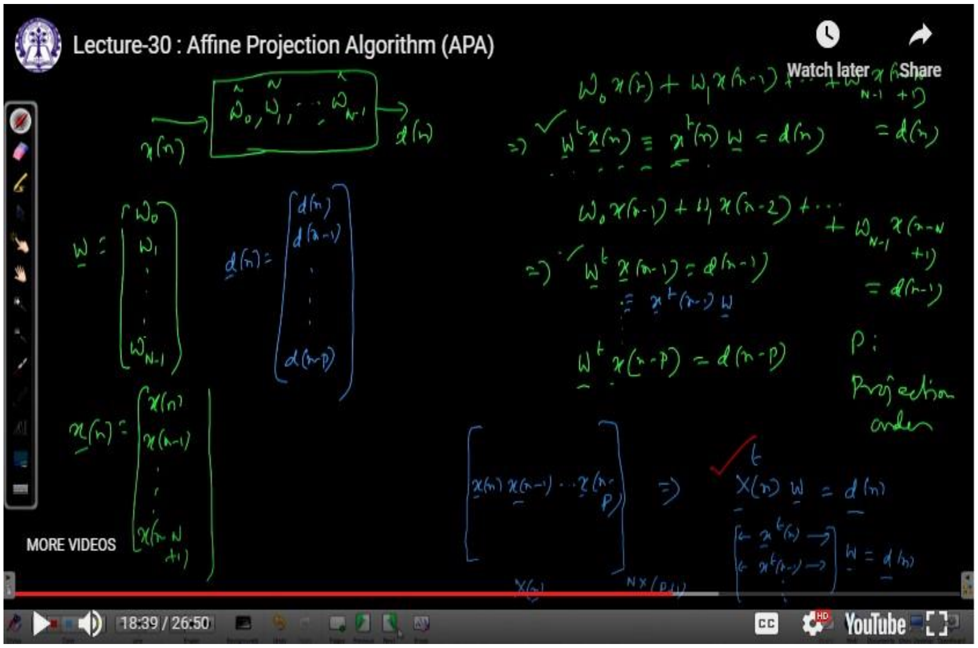
And if I define dN vector as current dN then previous dN dot dot dot plus dN sorry I made a mistake here this should not be this every xN vector is of length capital N, but how many I have I have x of N here x of N minus 1 here x of N minus 2 there the columns. So, last is xN minus p. So, it should be xN minus p all right this is my xN vector. So, this N cross p plus 1 N minus 0 N minus 1 up to N minus p should be p plus 1 this is the matrix. And you can see all these equations they imply this that xN transpose W should be dN vector because xN transpose W if it is a transpose of this first column will become first row.

$$\underline{X}^t(n)\underline{w} = \underline{d}(n)$$

So, column will become row means it will be a row x transpose N xN was column x transpose N is row then will be this way this second column will become second row. So, again x transpose N minus 1 xN minus 1 is column x transpose N minus 1 is row dot dot

dot dot and this W that is obviously, equal to dN vector because x transpose N W you have seen x transpose N W that is dN scalar x transpose this is equal to actually equivalent to like how I wrote equivalent. So, here x transpose N minus 1 W this two are equivalent. So, x transpose N minus 1 W will be dN minus 1 next guy and so on and so forth. So, it will become dN.

So, this is my equation this is the equation constraint. So, therefore, given WN I will try to find out one W which satisfies this that is which lies in the intersection hyper plane or intersection of all these hyper planes that itself can be straight line or a plane 3-dimensional plane or a 4-dimensional hyper plane or 5-dimensional hyper plane whatever that depends on N and P and all that.



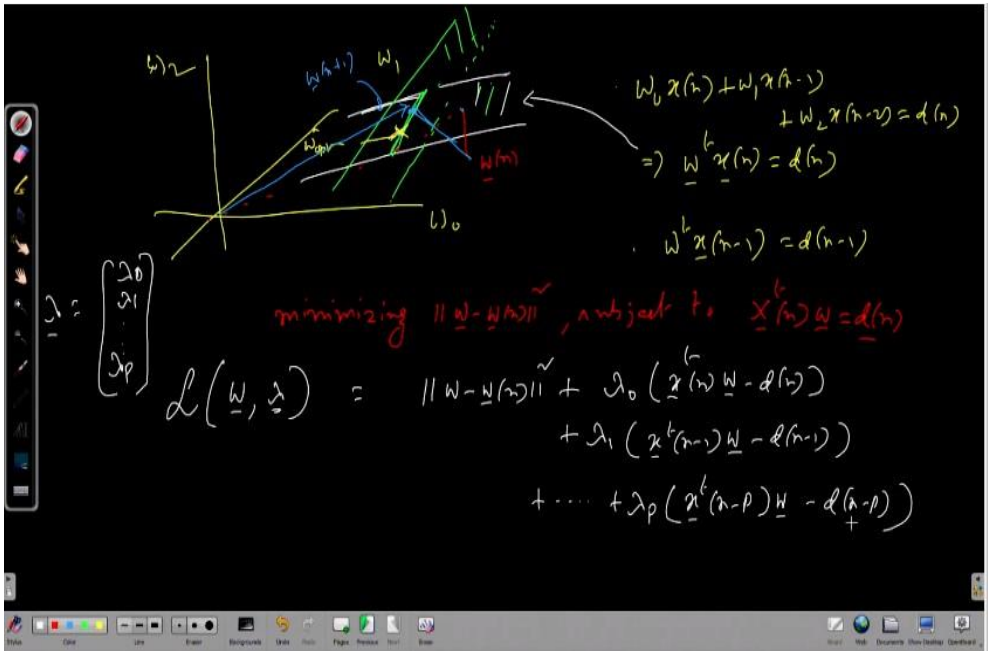
So, I will be projecting WN to W-in that intersection hyper plane that means my equation will be this I will be minimizing subject to this equation x transpose N W equal to dN. So, this will be in this is a constraint minimization. So, again I will bring Lagrange multiplier, but remember now I have not one constraint like in the case of NLMS just one straight line.

So, I had one constraint I have one here next here next to total P plus 1 constraint that is this should be satisfied simultaneously this should be satisfied simultaneously next will be satisfied simultaneously. So, I have one constraint here that will require one Lagrange multiplier say lambda 1 next is this that will require another one and like that. That means, I will now consider a general function W and you can call it lambda vector lambda vector has got all the Lagrange multipliers required. Wait a minute N N minus 1 fine. So, lambda 0 lambda 1 dot dot dot lambda P that is we should minimize this subject to first use lambda 0 that is constraint is xN transpose W minus dN then lambda 1 these are all Lagrange multipliers earlier I took only one now I have got so many because of these constraints.

Because you see if I differentiate this with respect to lambda 0 equate to 0 all other fellows will disappear only this will remain and that will equate to 0. So, that condition will be satisfied if I differentiate this with respect to lambda 1 then this quantity this will result and that will be equated to 0 others only disappear. So, this the second constraint next constraint will be satisfied so on and so forth plus dot dot dot dot last one is lambda P x transpose alright. So, this I have to minimize. So, this I can you can see I can write in this form first term remains as it is then if I say lambda transpose times xN we will verify minus dN W vector minus dN vector.

$$\underline{\lambda} = \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_p \end{bmatrix}$$

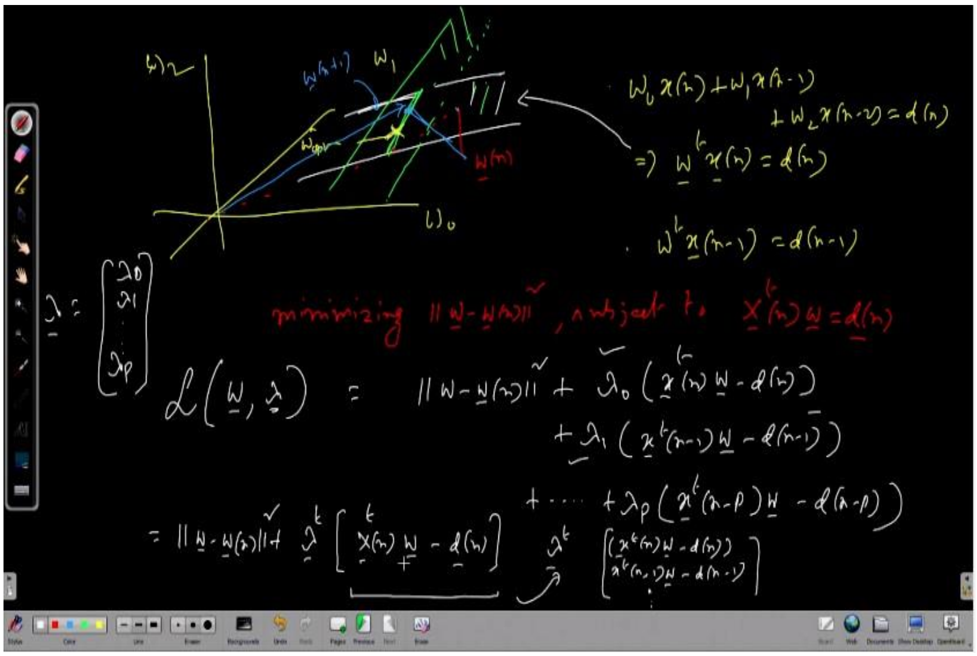
Let us see this we have seen in the previous page this is the constraint that is x transpose n W should be equal to dN we can even go into you know term wise x transpose n W minus dN.



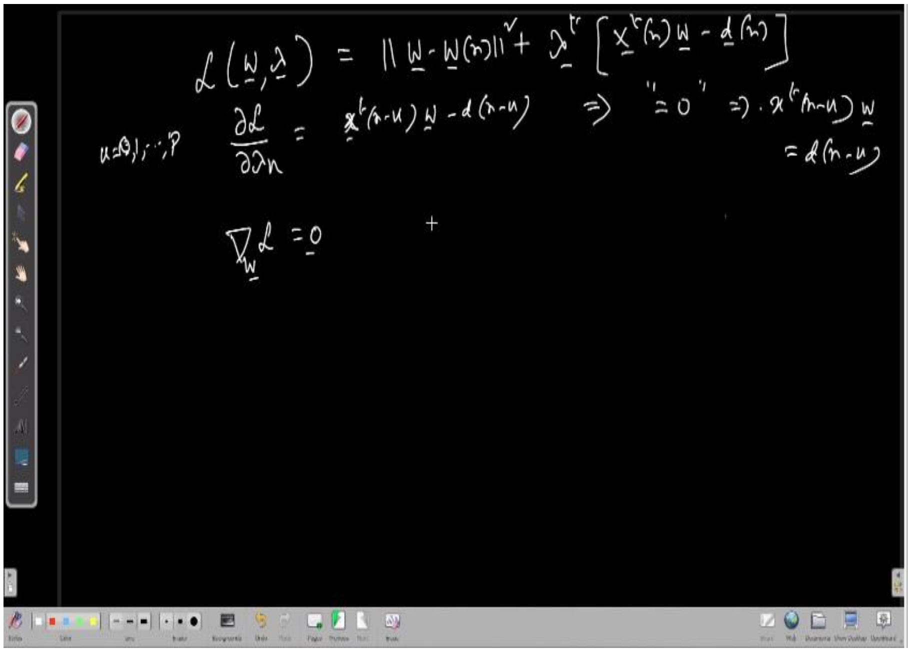
So, this part will be what this part will be a vector like this x transpose we have already seen in the previous page minus dN then x transpose N minus 1 minus dN minus 1 dot dot dot dot I already discussed a while ago. So, this is this vector matrix into column vector is a column vector minus again column vector. So, total is a column vector this is a row vector row into column is a scalar alright. So, this is xN transpose xN was a column vector x transpose N W this is a filter output that minus dN I do not know W.

So, I should have a W which satisfies this is equal to 0. Now if I have lambda transpose now if I have lambda transpose just I transpose it. So, it will be a row vector lambda 0 lambda 1 lambda p lambda 0 lambda 1 dot dot lambda p. So, lambda 0 will multiply this. So, that is what we will get this term this term plus lambda 1 times next.

So, we will get this term and dot dot dot dot. So, this is actually entire equation we can write like this compactly, but actually this is this the moment I differentiate L with respect to lambda 0 equate to 0 this only this much will remain and this will become 0. So, x transpose N W will be dN. So, that constraint will be satisfied if I differentiate this with respect to lambda 1 equate to 0 only this part will remain and equate to 0 means it will lead to x transpose N minus 1 W equal to dN minus 1. So, then the second constraint will be satisfied so on and so forth, but this whole expression I am writing in a compact manner like this all right.



So, with that I can go to the next page. So, first let me rewrite this is x transpose N minus K W N minus K and this we equate to 0. So, this constraint satisfied all right. So, separately we have to carry out these other partial derivatives also that is del W L and that is partial derivative of this with respect to each component of W put it in a stack and that again you have to I mean equate to 0 this is 0 vector now. So, this actually will lead to the value of lambda.



So, all these we will do in the next class. Thank you very much.