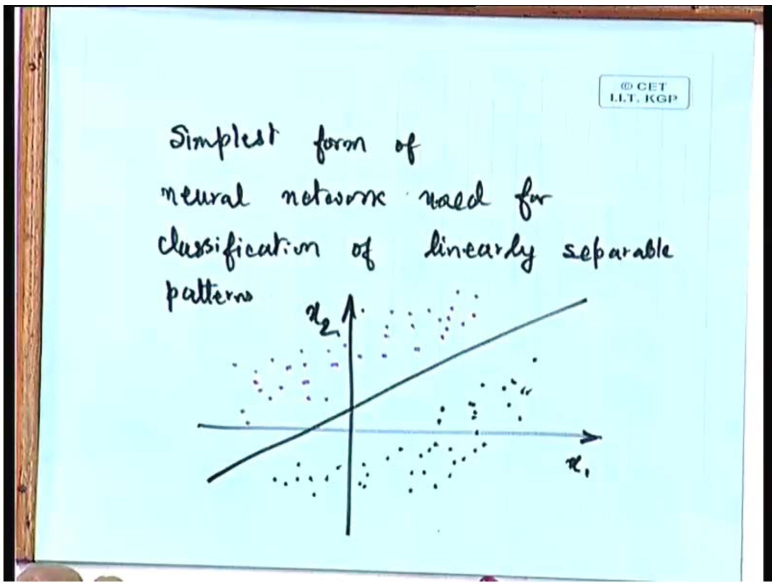
Neural Network and Applications

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

Lecture No -12
Single –Layer Perceptions

Single-Layer Perceptions, now in its simplest form actually, perception is a network that can classify linearly separable patterns. So, the simplest form of neural network.

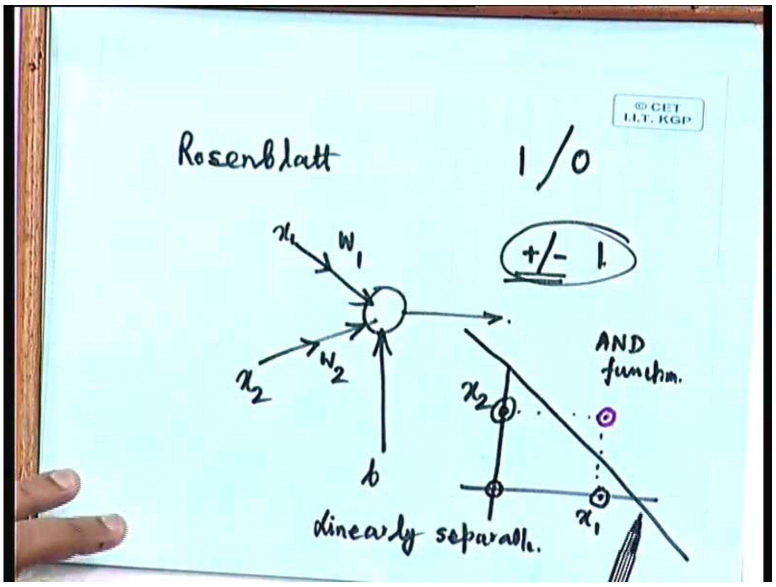
(Refer Slide Time: 01:17)


Simplest form of neural network used for classification of linearly separable patterns

That is, used for classification of linearly separable patterns, what is meant by linearly separable patterns that means to say that, let us consider a two dimensional space again our most simplified example, that supposing we have the space as x 1 and x 2. Two input perception let us say and we have got a set of patterns which are all these things, so these are the set of one group of patterns and supposing, these are the set of another group of patterns.

Now, it is very clearly linear is separable because, we can find one straight line, which is going to distinguish between these category of patterns from these ones, so they are indeed linearly separable. So, this is a two dimensional case, we extended it to three dimension there, we should be able to pass a plane that should separate the patterns giving an output a plus 1 from the patterns that give output of minus 1, we extend it to dimensions greater than three, then it becomes a hyper plane that has to separate the minus 1 patterns from the plus 1 patterns, now it was shown by Rosenblatt.

(Refer Slide Time: 03:20)



In fact, Rosenblatt is one of the key researchers, in whose name one can associate the perceptrons that if you have a linearly separable class of patterns then a perceptron converges, now a perceptron units simplest form would look like this let say again two input perceptrons. Let us say that we have got input as x 1 x 2, the synaptic weights as w 1 and w 2 because we are considering only one neuron, that is why I am not putting any other suffix saying it has w 1 1 and w 1 2, I mean there is only one neuron.

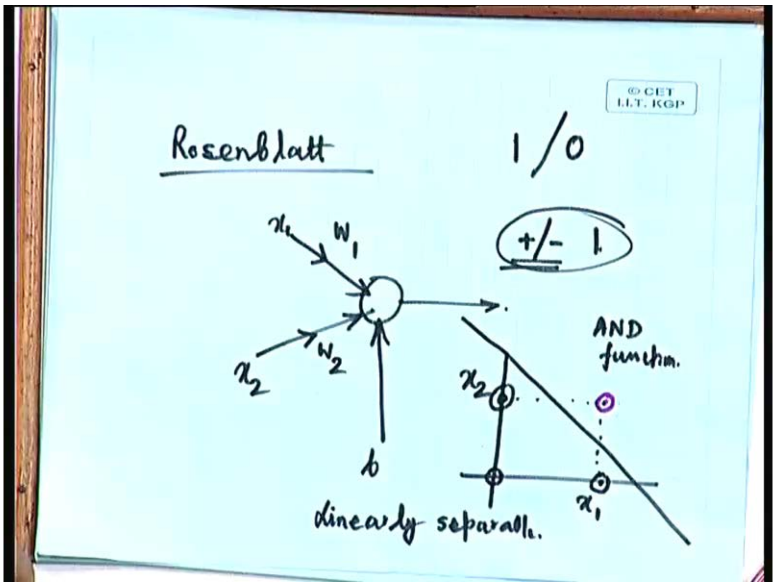
And naturally there will be some bias, which will be there bias b and there will be an output and let us say that it is giving us a activation of plus and minus 1. Now, let us consider some very small examples, let us say that we are going to realize n and get out of it and n get means that in a two dimensional space x 1 here, x 2 here, when x 1 and x 2 both are going to be 0 then the output will be equal to 0 or if, we take the activations to be plus 1 and minus 1.

Then, we should say that it should lead to minus 1 or let us say that why to consider plus minus 1. Let us say that it is 1 or 0; that means to say that if it exceeds a threshold it is equal to 1 greater than or equal to threshold it is equal to 1. And if it is less than the threshold then it is equal to 0, so when x 1 is equal to 0 and x 2 is equal to 0 then the output is equal to 0.

When x 1 is equal to 1 and x 2 is equal to 0 then also the output is 0, when x 2 is equal to 1 and x 1 is equal to 0; that means to say this point then also the output is equal to 0, but when x 1 and x 2 both are equal to 1 then the output is going to be 1. So, this is a different class, so that is why I indicated by a different color, so this is the simplest, that is the AND function that we can consider.

Now, is it a linearly separable pattern, very clearly because we can consider a line like this which is going to separate the patterns giving response equal to 0 from those patterns, which are giving response equal to 1. So, this is a linearly separable and naturally we can solve this problem using a two input perceptron, can solve this problem. Let us take OR function.

(Refer Slide Time: 07:00)



Again using the same model OR function means that when we have x 1 and x 2 both equal to 0 output is 0, but for all the other three cases will be having one, one and with x 1 x 2 both equal to 1 also the output will be equal to 1. In this case is the problem linearly separable very clearly yes, we can find a line that distinguishes the 0 patterns from the 1 patterns, so this is OR function and this is linearly separable and we can solve this very easily using a two input perceptron.

Now, let us take an Ex-OR function, Ex-OR function means what again take the x 1 x 2 axis, with x 1 and x 2 both equal to 0 the output is equal to 0, when x 1 is equal to 1 and x 2 equal to 0 output is 1, when x 1 is equal to 0 x 2 is equal 1 output is 1 and when both of them are equal to 0 then the output is equal to 0, so this is 0. Now, is this linearly separable why.

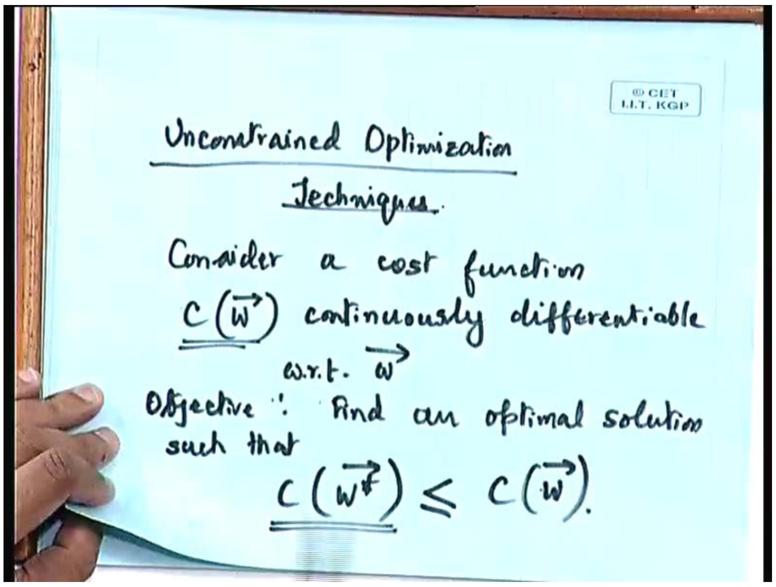
Student: ((Refer Time: 08:40))

You cannot pass a straight line that distinguishes the 1 patterns from the 0 patterns, I can try to think of any number of straight line not this one, not this one, not this one, none of the straight lines would be able to clearly distinguish that one side of the line, when we having 0 patterns one side of the line 1 patterns it is never possible. So, Ex-OR problem is not linearly separable, so we cannot solve this problem using a two input perceptron.

We have to do something else, to input perceptron or rather to say even if we extend this Ex-OR to multi dimensions, still since it is not linearly separable, it is not possible for us to form a solution out of perceptrons there we need actually the multilayered connections, but more about that later on. Now, let us go somewhat in to the theory of the perceptrons, because we once made a statement that it was shown by Rosenblatt that if you can find linearly separable class of patterns, if we train a perceptron a single layer perceptron with linearly separable patterns, then it converges.

I mean, ultimately it should be able to in an m dimensional space ultimately it will be able to find the hyper plane that separates the 1 patterns from the 0 patterns, so convergence is says is guaranteed, but let us go over to that convergence theory. So, in order to go in to the convergence theory, let us consider the unconstrained optimization technique.

(Refer Slide Time: 10:47)



© CET I.I.T. KGP

Unconstrained Optimization Techniques.

Consider a cost function $C(\vec{W})$ continuously differentiable w.r.t. $\vec{W}$

Objective: Find an optimal solution such that

$$C(\vec{W^*}) \leq C(\vec{W}).$$

Because ultimately we have to see that at what point it leads to the minimum error, so it is the unconstrained optimization through which we will try to explain the convergence aspects. Now, when we train a network naturally what we are measuring is that, we are finding out the difference between the actual response from what the target response says because in a training pattern, we are feeding the inputs as well as target output, so the difference that exists between the target output and the actual output that is the error.

And we are taking some measure of the error, may be the sum of squares and thinks like that which will give us some error cost function and naturally our objective will be to minimize the cost function. In fact, we had discussed that in our initial series of lectures itself, we had discussed about that but we will just formulate it more into an integrated theory. Now, let us consider a cost function, consider a cost function, now cost function is; obviously, a function of the free parameters.

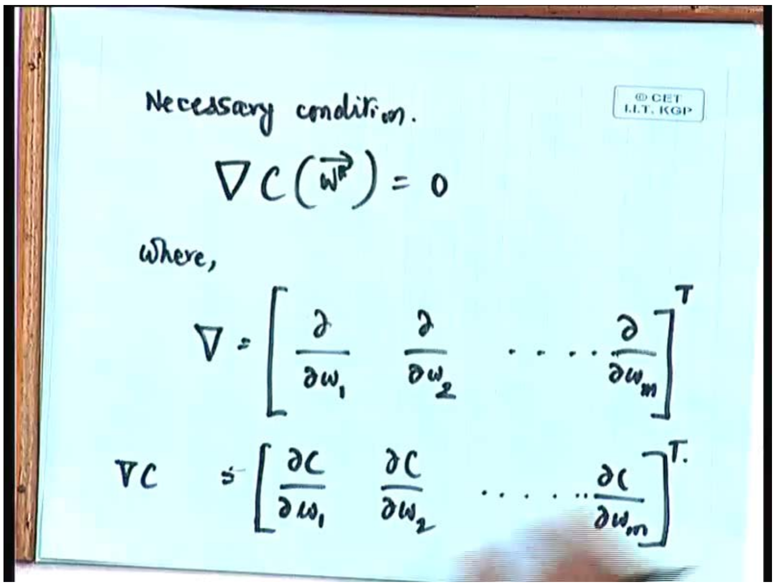
So, if we call the cost functions C, we should write as an argument of C the w vector, the weighed vector or the free parameter vector whatever you call and let us consider also that this cost function is continuously differentiable, this is very important that the cost function that we consider a C w is continuously differentiable with respect to the w vector. In fact, if it is not continuously differentiable then it leads to a problem then we have, problem in adapting any steepest decent approach or any other optimization approach, if it is not continuously differentiable.

So, that is the basic assumption that whatever cost function we consider is indeed continuously differentiable, so what our objective is to find, so our objective is to find an optimal solution of what because we are going to adjust what, we are going to adjust the weights; that means to say this w is something that we are going to adjust. So, optimal solution we are going to find such that let us say, that we arrive at some weight w star.

Let us say that we can we have found out, some weight w star such that C of w star is less than or equal to C of w, if we can have that then we can say that the C w star is the optimal cost or other w star that is the argument of this C w star that w star is going to be the optimal weight, which we are going to achieve. Now, what is the necessary condition of that the, now naturally we are not considering any I mean over simplified case, we should consider now a very general case that this w will be typically and m dimensional vector.

So, in terms of any m dimensional vector also we should say that it is gradient in the multi dimensional space, it is gradient at that point should be equal to 0, so; that means to say that if we take gradient of this cost function, that should lead to 0, if this is the point of optimality if C w is the minimum cost that we can find out from this system.

(Refer Slide Time: 15:24)



Necessary condition.

$$\nabla C(\vec{w}^*) = 0$$

Where,

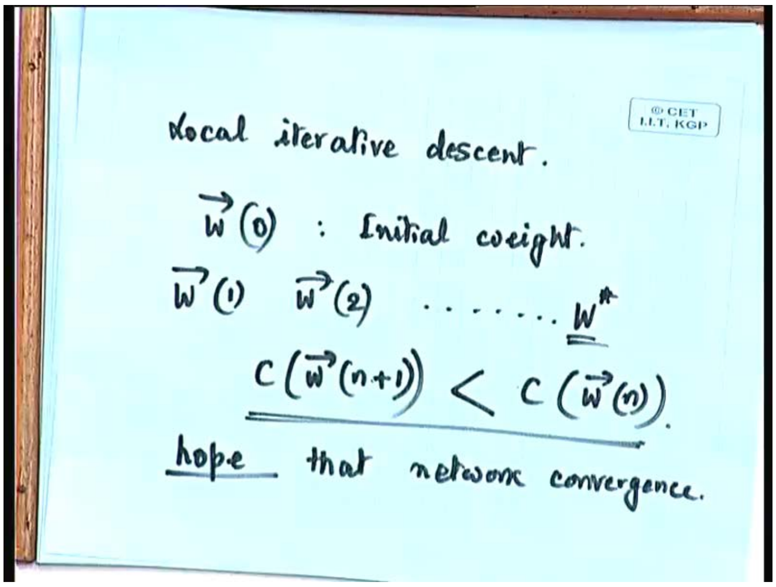
$$\nabla = \left[\frac{\partial}{\partial w_1} \quad \frac{\partial}{\partial w_2} \quad \cdots\cdots \frac{\partial}{\partial w_m}\right]^T$$

$$\nabla C = \left[\frac{\partial C}{\partial w_1} \quad \frac{\partial C}{\partial w_2} \quad \cdots\cdots \frac{\partial C}{\partial w_m}\right]^T.$$

Then, the necessary condition for that is grad of C w star is equal to 0, so this is the necessary condition where, what is the gradient operator, the gradient operator will be defined as del del w 1 del del w 2 up to del del w m, where we are considering m dimensional vector and because we are writing it as a row vector, now we just indicated with a transpose. So, this is the gradient operator so; that means to say that when we talk of grad C.

Grad C I am just not writing this w every time then we talking of grad C as this vector dou C dou w 1 dou C dou w 2 dou C dou w m this transpose. So, the question is that what is that we are going to find out, so the approach that we follow for this is the local iterative descent.

(Refer Slide Time: 16:50)



Local iterative descent.

$\vec{w}(0)$: Initial weight.

$\vec{w}(1)$ $\vec{w}(2)$ .... $\underline{\underline{w^*}}$

$$\underline{\underline{C(\vec{w}(n+1)) < C(\vec{w}(n))}}.$$

<u>hope</u> that network convergence.

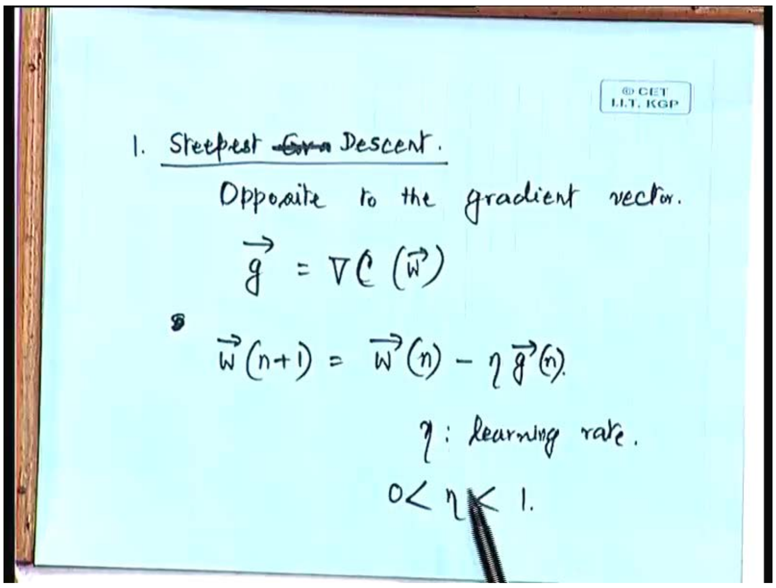
And by local iterative descent, what we mean is that we start with sum initial weight vector assumption w 0, the reason why I am putting as an argument I am now going to put a number, that number will indicate the iteration number. So, w 0 is the weight that it will have at iteration number 0; that means to the starting weight or the initial weight, so this is initial weight and In fact, in all the cases, we are going to have some guess of the initial weight.

In fact, it does not matter, we can have any arbitrary guess because if the solution, if the local iterative descent algorithm works then with any arbitrary weight combinations also, we should ultimately reach the point of minimum that is what we will looking for. So, from this, so starting with this w 0 we are going to get a sequence of weight vectors, which we are going to write as w 1 w 2 etc continues, I mean we do not now that at what point we are going to find out the minimum.

Ultimately we are going to find the w star somewhere and if we can find out the w star that is where we are going to stop, but a question is that what must be ensured for a convergence is that the cost function of w at the n plus 1 th iteration, what at n plus iteration, the cost that we are going to get must be less than the cost that we had at the iteration n. So, if this is valid then, we can say that the network will ultimately converge or let us hope that network convergences.

So, I purposefully write the word hope, saying that sometimes our hope may not be fulfilled and under what condition, the hope does not get fulfilled that also we have to find out. So, now we are talking about the unconstrained optimization, so we are going to just describe a few standard techniques in order to have the unconstrained optimization and what are the standard techniques, let us go one by one on that.

(Refer Slide Time: 19:47)


1. Steepest ~~Gra~~ Descent.

Opposite to the gradient vector.

$$\vec{g} = \nabla C(\vec{w})$$

$$\vec{w}(n+1) = \vec{w}(n) - \eta \vec{g}(n)$$

η : learning rate.

$0 < \eta < 1$.

So, the first approach that we had already described in one of the previous lectures is the steepest gradient, so this time I can go fast and is steepest descent, so this time I can go little fast because, the basic theory is already known to you. So, steepest descent as you know is traveling in a direction opposite to the gradient vector, so you have to travel opposite to the gradient vector or rather to say, we have to travel downhill because the gradient will be indicated uphill.

And that is why, we have to travel opposite to that that is downhill and always downhill movement is something about which we should be careful, you know when we are climbing down, a hill that time if we can take two different strategies, one strategy is that we just make our steps, so cautious that ultimately we are going to reach the point of minimum after taking quite long time may be, that we will take few hours to reach the point of minimum. The other approach could be is that we start running, while traveling downhill and you now that running down hill is quit risky.

If you cannot control your steps, then there is a risk that you will not only reach the, it will go through the point of minima and you give will even overshoot the point of minima and start climbing once again. So, that is something that you have to avoid, so it is a downhill travel that we have to understand, anyway so direction opposite to the gradient vector.

Now, what is the gradient vector, gradient vector I said is del C as a function of w vector and instead of writing this del C w every time, I am just going to denote it by a new notation g vector, so g vector is nothing, but these gradient vectors, so g vector is one and the same as this followed. So, accordingly the steepest descent can be formally described as the one, which we have already discussed.

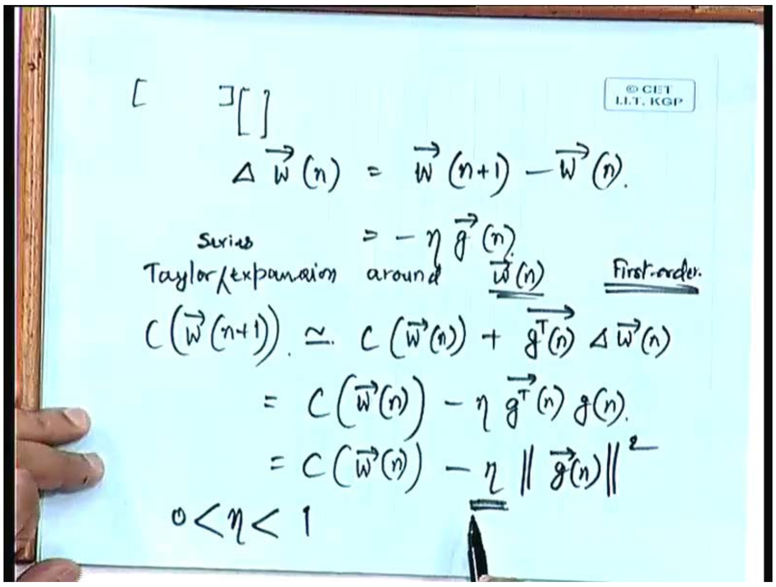
That w at the n plus 1 th iteration will equal to the w at n th iteration minus, minus what remember eta g n, where eta is the learning rate remember that we discussed this earlier. And somebody said delta, delta means it is the gradient effectively, it is the gradient at multiplied by the learning rate and the learning rate should be typically small. In fact, learning rate typically should be a quantity that we define between 0 and 1.

It is normally defined in terms of a fraction, so that effectively you subtract a small vector from this, yes any question.

Student: ((Refer Time: 23:21))

Yes, we can get stuck in a local minima and that is one problem that one has to tackle differently, in fact, local minima is tackled by a method which is called a simulated annealing and we will be discussing about that later on. In this case the simplest of assumption that we make is that we are having a convex hyper surface, where a global minima exist. Now, the weight correction that we are going to apply on this is delta w n.

(Refer Slide Time: 24:01)


©CET I.I.T. KGP

$$\Delta \vec{w}(n) = \vec{w}(n+1) - \vec{w}(n).$$

$$= -\eta \vec{g}(n).$$

Taylor series expansion around $\underline{\vec{w}(n)}$ First-order.

$$C(\vec{w}(n+1)) \simeq C(\vec{w}(n)) + \vec{g}^T(n) \Delta \vec{w}(n)$$

$$= C(\vec{w}(n)) - \eta \vec{g}^T(n) g(n).$$

$$= C(\vec{w}(n)) - \eta \|\vec{g}(n)\|^2$$

$$0 < \eta < 1$$

That can be expressed as w this is the difference of the weight, which can be expressed as w n plus 1 minus w n, which is according to this equation, this is the change in weight, change in weight is simple at n plus 1 th iteration whatever w was there from that we have to subtract the weight, that was there and the n th iteration. And according to this expression wn plus 1 minus wn is equal to minus eta gn, so we can simply write this one as minus eta gn.

Now, what we have to know our starting point was that we must know the cost at the n plus 1 th iteration that is what we are looking for is in it. So, we can express the cost at n plus 1 th iteration, is it possible for us to express the cost at n plus 1 th iteration in terms of the cost that we had at n th iteration. And in terms of these deltas, it should be possible using…what I can hear that, somebody is whispering the technique that is to be used. What is it can you speak out.

Taylor series expansion that is correct, I mean people where mentioning about Taylor series not very confidently, but. I mean, it is the Taylor series expansion that we have to carry out for this Cw n 1, so we make Taylor series expansion around wn to approximate Cw n plus 1, so, Cw n plus 1 can now be simply we make a first order assumption. So, we simply make a first order assumption of Taylor series expansion

So, that we can write Cwn plus 1 as Cwn plus yes gradient, gradient of w because we are making Taylor series expansion around wn, so the gradient of wn what is the gradient of wn gn, gn and this should be multiplied by yes and what is that in this case, the change what is the change delta wn, delta wn is a small change that we are making, so it is gn times delta wn. In fact, delta wn is a vector naturally, because w is a vector and you are subtracting these two vectors.

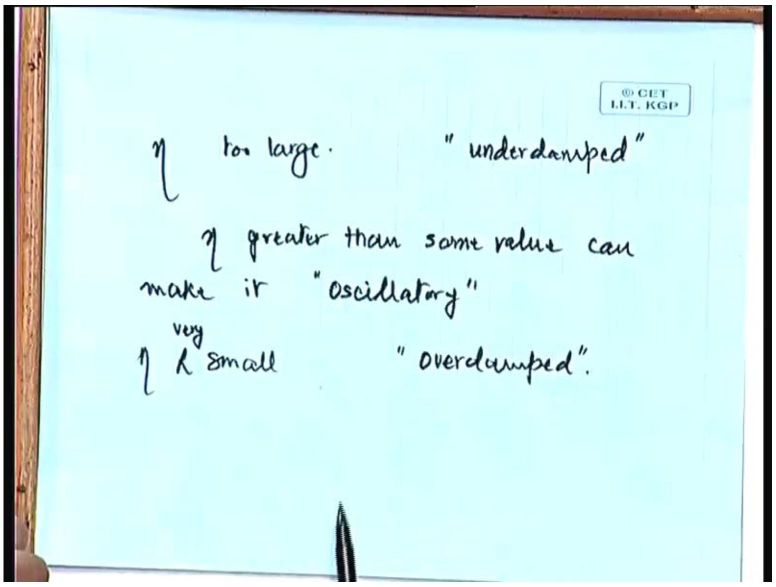
So, delta wn is also a vector and if we express delta wn as a column vector, In order to multiply this by gn we must express gn as a row vector, so that is why the expression that we should write down is that it should be g transposing, so that ultimately we get the dot product of this g vector, g transpose and this delta w n. So, that should lead to a scalar quantity that we will add to this Cwn, this is simply that we will get by the first order Taylor series expansion.

And this is equal to Cwn minus, minus what because we had already got this delta wn as eta gn is in it and eta is a scalar quantity, so eta can be taken out and then what we have is g transpose n times gn and what is g transpose n and gn norm of gn square. So, this is equal to Cwn minus eta norm of gn square this is very clear is in it, because what you are having that you are multiplying g transpose with the g vector.

So, if the elements of these are g 1 g 2 g 3 etc and here also g 1 g 2 g 3 etc, you multiply the two you get g 1 square plus g 2 square plus g 3 square up to gm square, so here this is norm of that. So, we get Cwn minus eta gn square, now what we are looking for, we are saying that we must have Cw i wait every iteration the cost must decrease is in it. Now, is it decreasing in this case, definitely decreasing because eta by our choice is a quantity that is lying between 0 and 1

So, eta is positive and what is gn square gn square is also positive, so, this is a positive quantity, so that whatever cost we had at n th iteration at the n plus 1 th iteration it decreases in the cost. But, what about the magnitude part of the cost because there is one important parameter that plays a role in this and this eta.

(Refer Slide Time: 30:10)



η too large. "underdamped"

η greater than some value can make it "oscillatory"

η very small "overdamped".

You make eta too large, so eta too large leads to what yes it will cross the point of minima and travel further, if you make eta very large than it will cross the point of minima and it will go beyond that you will miss the minima. And then again once it comes back to one of the point, it will again try to go in the negative to the gradient, minus of gradient direction, so it will again try to go to the minima and possibly this time also it will cross that point, so it will followed by overshoots and undershoots.

So, if eta is too large then we will come back to whatever point you have, so if eta is too large then it leads to an under damped case, so under damped means that it is going to have overshoots and undershoots and In fact, it is a and it may be even oscillatory. So, eta greater than certain value, greater than some value can make it oscillatory, whereas if eta is small in that case the response is going to be, if it is very small, if it is too small or very small then it is it is going to be over damped and over damped means you will reach the convergence.

You will reach the point of minima, but you will take more number iterations, your number of n in order to reach w star will be quite large. So, any questions that somebody want to ask yes please.

Student: ((Refer Time: 32:13))

It is decreasing, I am not denying on that, but thing is that you can if you are taking the mode of that then you are, but

Student: ((Refer Time: 32:47))

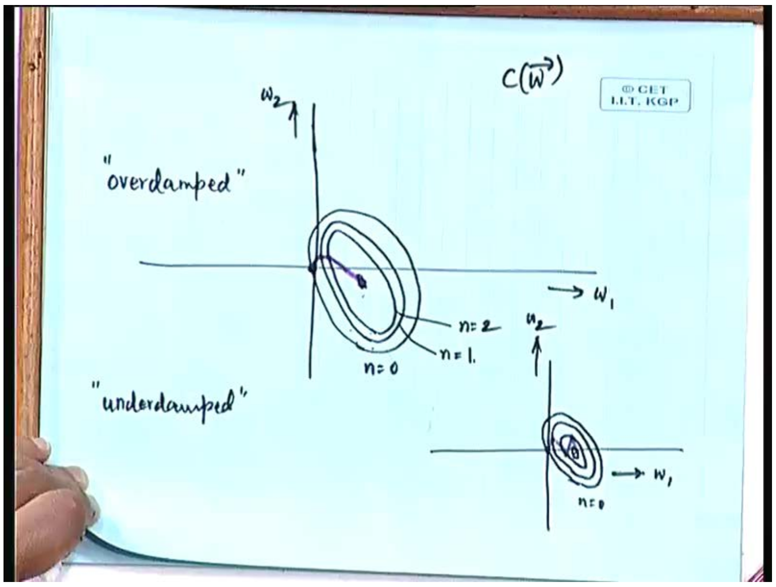
So, will we will come back to this point, but even going by yes higher order terms definitely come in to play yes, that is that is very correct observation yes, that the higher order terms, which we had neglected in this process will play a role in making it unstable. So, that is why it is very important to control this eta, because if you make eta on the smaller side then you may increase the number of iterations, but there is no risk of missing the minima and going in to the under damped case.

Student: ((Refer Time: 33:40))

No, we are normally keeping eta, actually there are different types of learning philosophies, which people adapt during the neural network training in fact, it is a good at the question has already come to your mind that whether, we are keeping the eta constant throughout. There are some learning strategy is which people consider that to start with they take larger etas, so that initially when you do not have the risk of crossing the minima you travel fast.

And as you go more towards the point of minima, you gradually decrease the eta, so such learning strategy is do exists, so in such kind of strategies it is often easier to control the convergence. If eta is made like that, but so, far let us assume to start with, let us assume that eta is something, which we are keeping as a constant throughout the training process. Now, in this case what is the picture that we are getting, what is the representation that we can imagine out of this type of a cost decreasing philosophy.

(Refer Slide Time: 35:09)



Again let us try to visualize the scene with a simplest two dimensional case, so let say that we have got as a free parameter in fact, since we are bothered about the cost as a function of w, we must consider the elements of this w vector and just as a very simple case, we take a perceptron network, which is having only two free parameters w 1 and w 2. Now, the cost that is there the cost will be; obviously, a function of w 1 and w 2 and it may be that for different combinations of w 1 and w 2 the cost may be the same.

That supposing it is quadratic cost function as we are mostly taking, in that case the quadratic cost function would have a contour like this. Supposing, for n equal to 0 this is the cost contour that we have got and cost control means that if we take any point on this contour then the cost of that is going to be the same, so this is the cost contour. And never know this could be our starting point, if we take this to be our starting point that means to say that we are starting with w 1 equal to 0 and w 2 equal to 0 from that point.

Now, for n equal to 1 because contour may be like this, for n equal to 2 the cost contour could be like this, for n equal to 3 it may be like this, like that ultimately there will be a point where you reach the minima and that is by you are want to go. Now, in the case of the over damped, in case of over damped response or rather if we control the eta if we make eta too small, so that the response is over damped then we will be traveling somewhat like this.

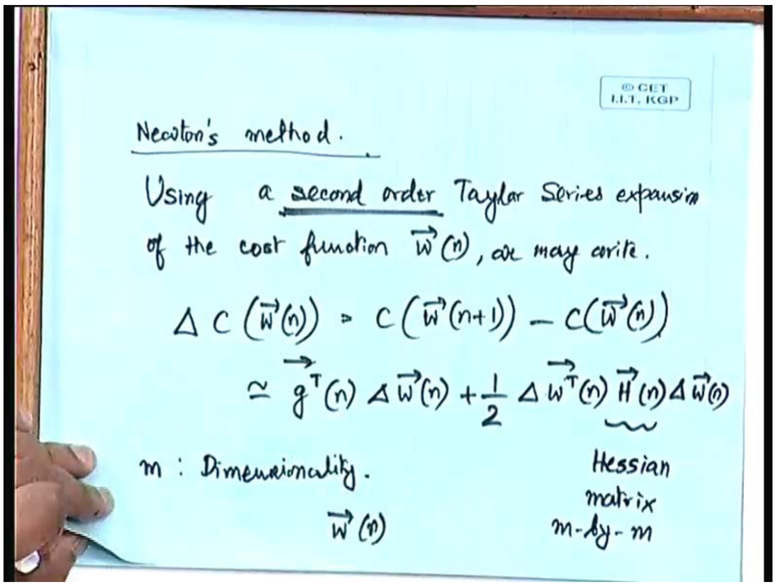
That supposing this is our starting point; that means to say with w 1 equal to 0 and w 2 equal to 0, with n equal to 1 we may reach this point, with n equal to 2 we may reach this point and supposing this is the point where we want to go. So, ultimately we are going to reach this point of minimum by traveling this way, so here this is the typical case of over damped, now the response could have been under damped also and in case it was under damped then what would have happened.

In that case it would have traveled in a zigzag fashion, so may be that here, so here if we take again a two dimensional space w 1 and w 2 and if these are the contours that we imagine. It is a contour for n equal to 0, for n equal to 1, for n equal to 2 etc. so, on. Now, in this case we would have reached from this point we might have gone here, here, her, here, like this it would have traveled in a zaggy fashion, it would have in a zaggy path in order to ultimately reach the point of minimum.

So, here it is a smooth travel, here it is a zaggy travel, so there are two thinks in it, but here we may be able to reach faster as compared to what we have done in the case of over damped thing. So, this is the simplest form of approach that we take, where we take only the first order assumption and by taking first order assumption, we can see that first order assumption; obviously, guarantees a convergence no doubt, but because of the negligence of the higher order terms.

The presence of eta can play a nasty role, you make eta too large it can lead to oscillatory or too much of under damped response, which a something that we do not want. Now, we discuss about a second methodology of unconstrained optimization, where we do not neglect the second order term, we consider the second order term of Taylor series expansion also and that is done using what is called as the Newton’s method.

(Refer Slide Time: 40:22)



Newton's method.

Using a $\underline{\text{second order}}$ Taylor Series expansion of the cost function $\vec{w}(n)$, we may write.

$$\Delta C(\vec{w}(n)) = C(\vec{w}(n+1)) - C(\vec{w}(n))$$

$$\simeq \vec{g}^T(n)\,\Delta\vec{w}(n) + \frac{1}{2}\Delta\vec{w}^T(n)\,\underset{\sim}{\vec{H}(n)}\,\Delta\vec{w}(n)$$

m : Dimensionality.

$\vec{w}(n)$

Hessian matrix

m-by-m

So, using Newton's method we can follow unconstrained optimization, so here the basic idea is to minimize the quadratic approximation of the cost function, so using a second order Taylor series expansion around wn, we may write what here, if we calculate the difference of the cost. We express the difference of the cost delta C wn, delta same is the difference of the cost and the difference of the cost will be what C of wn plus 1 minus C of wn.

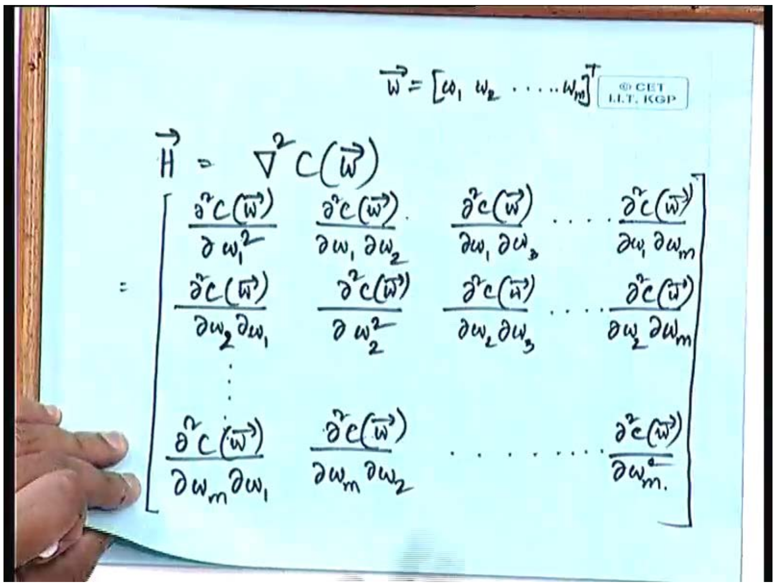
So, this is cost difference and this cost difference, in fact, earlier case also it was a cost difference only, earlier case it was delta C wn the one that I am writing just now. In the earlier case delta w delta C wn was, Cwn plus 1 minus Cwn which was simply the first order terms. So, gTn delta wn was the only term that was there in the delta C, but in this case this gT wn will any way remain and with that we will be having a second order term, so what is that. So, we can approximate this again, approximation sign will be there because after all it is again a second order approximation.

We are still neglecting the higher order things, so this can been expressed as the first order term which is simply gTn delta wn, the earlier expression that we have got plus the second order term in fact the second order term is expressed in this manner. It is expressed as half of delta wTn into Hessian matrix Hn times delta wn, so I will just write down the expression, so that you can easily understand that why this half has come about and in fact, this is a second order derivative that we are taking is in it.

After all this is a second order derivative and if we are taking a second order derivative of the quadratic cost function, it can be shown mathematically that ultimately it leads to this. So, I am not going into the derivation part of the mathematics over here, but simply writing down the second derivatives by this type of an expression, but the second order derivatives have been embedded in this matrix, where this matrix is called as a Hessian matrix, in fact, because we are taking the dimension of the network to be m.

m is the dimensionality of the network; that means to say m input, so in this case the Hessian matrix will be of size m by m, so, it will be a m by m Hessian matrix that we are evaluating at wn, so it is a Hessian matrix evaluated at wn. And what is the form of the Hessian matrix, the form of the Hessian matrix will be as follows.

(Refer Slide Time: 44:42)



$$\vec{w} = [w_1\ w_2\ \ldots\ w_m]^T$$

© CET I.I.T. KGP

$$\vec{H} = \nabla^2 C(\vec{w})$$

$$= \begin{bmatrix} \frac{\partial^2 C(\vec{w})}{\partial w_1^2} & \frac{\partial^2 C(\vec{w})}{\partial w_1 \partial w_2} & \frac{\partial^2 C(\vec{w})}{\partial w_1 \partial w_3} & \cdots & \frac{\partial^2 C(\vec{w})}{\partial w_1 \partial w_m} \\ \frac{\partial^2 C(\vec{w})}{\partial w_2 \partial w_1} & \frac{\partial^2 C(\vec{w})}{\partial w_2^2} & \frac{\partial^2 C(\vec{w})}{\partial w_2 \partial w_3} & \cdots & \frac{\partial^2 C(\vec{w})}{\partial w_2 \partial w_m} \\ \vdots & & & & \\ \frac{\partial^2 C(\vec{w})}{\partial w_m \partial w_1} & \frac{\partial^2 C(\vec{w})}{\partial w_m \partial w_2} & & \cdots & \frac{\partial^2 C(\vec{w})}{\partial w_m^2} \end{bmatrix}$$

That Hessian matrix will be the Laplacian of or the second derivative of this cost function and this is expressible as follows, del square C w del w 1 2, so this is the second derivative with respect to w 1 because a w 1 is the first element of the weight vector. So, weight vector is again w 1, w 2, up to w m transpose, so this is the weight vector, so at first we have to take the second derivative with respect to w 1.

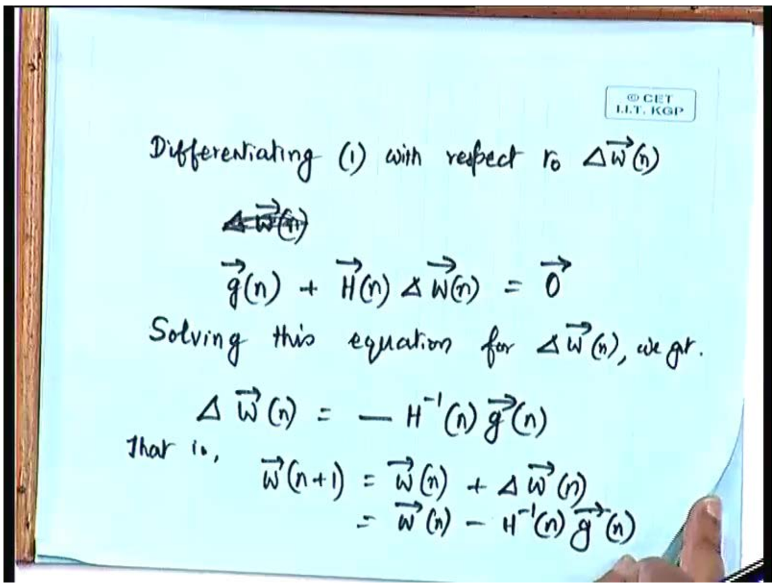
What will be the next one, del square by del w 1 del w 2, what is the next term, del w 1 del w 3 and the last term will be del square C w del w 1 del w m because we are considering m by m matrix. So, this is the first row and what will be the second row correct w 2 w 1 yes, what will be here correct, very correct, here it will be w 2 w 3 and then the last row will be here wm w 1 wm w 2 and the last term will be wm square.

So, this is the Hessian matrix write, so it involves all the second order terms second order derivative terms and if you take a second order derivative then this is what it will result, so you have a pre multiplication of the Hessian matrix in this case is pre multiplied by w transpose and the post multiplied by w n. In fact, this can be easily derived from matrix theory that if we what taking the second derivative this is what results for the quadratic function.

And in this case, this is very important that if we, now take this delta w Cn, delta Cwn if we take then what we are trying to achieve, we are trying to minimize the cost function is in it. So, if we now differentiate the delta term with respect to wn then the change delta Cwn will be equal to 0, the change will not happen when this delta w is I mean delta C differentiated with respect to w that becomes equal to 0, so if we differentiate this whole expression delta C is equal to this whole thing.

If we, now differentiate with respect to w, so let me put some number to this equation, so here it is delta Cwn equal to this and we put this as equation number 1then differentiating equation number 1 with respect to delta w.

(Refer Slide Time: 49:14)



Differentiating (1) with respect to $\Delta\vec{w}(n)$

~~$\Delta\vec{w}(n)$~~

$$\vec{g}(n) + \vec{H}(n)\Delta\vec{w}(n) = \vec{0}$$

Solving this equation for $\Delta\vec{w}(n)$, we get.

$$\Delta\vec{w}(n) = -H^{-1}(n)\vec{g}(n)$$

That is,

$$\vec{w}(n+1) = \vec{w}(n) + \Delta\vec{w}(n)$$
$$= \vec{w}(n) - H^{-1}(n)\vec{g}(n)$$

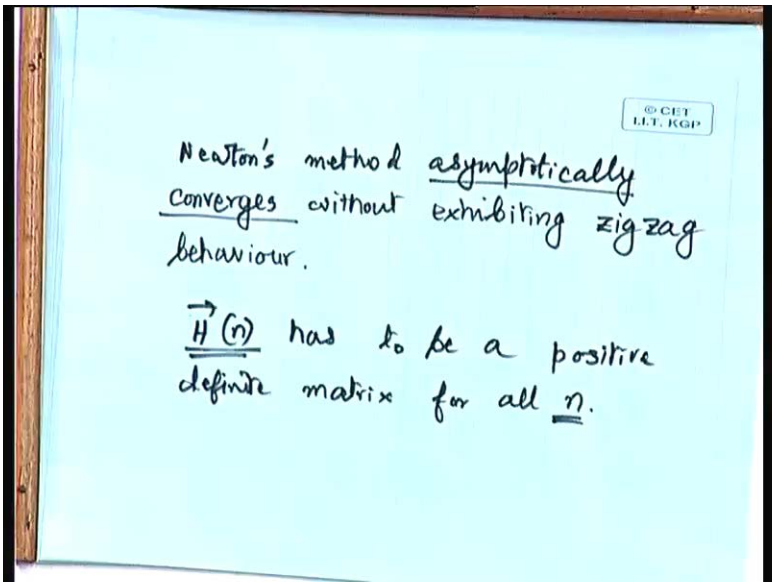
So, this one if you differentiate with respect to delta wn, delta w rather then what you can expect, you get the gTn terms as say and as it is. In fact, if you differentiate this it will be not the gTn, it will be gn term which will come. So, the matrix differentiation I think, I will make it more lucid in the next class by presenting the matrix differentiation theory, if you what not a tune with the differentiation that we are carrying out that moment.

If you differentiate this then the first term will result in gn and second term will result in what can anybody guess, the second term will be Hn multiplied by delta wn this also can be by shown matrix differentiation. So, we are differentiating everything with respect to delta w, so that is what you remember, so then we get this expression gn plus Hn delta wn and this must be equated to 0, in fact gn is a vector Hn delta wn that also leads to vector m dimensional vector.

So, this 0 that we are writing is going to be a 0 vector m dimensional 0 vector, so that is why I am putting the vector notation here and solving this equation for what, we are going to solve it for delta wn. So, we get delta wn is equal to minus of H inverse n correct gn, so; that means to say, so because we have already got this as delta wn, so we can write wn plus 1 is equal to wn plus delta wn and that is equal to wn minus H inverse n gn.

I simply substitute for this delta wn, I simply substitute the one that we have got, so this is what we get as this, now again the million dollar question is that is it really converging, In fact, it has been shown that Newton's method converges asymptotically and in fact, here because we are taking this second order term, the convergence in this case is guaranteed and it converges asymptotically and without exhibiting any zig zagging behavior, so Newton's method.

(Refer Slide Time: 53:19)



Newton's method asymptotically converges without exhibiting zigzag behaviour.

$\underline{\vec{H}(n)}$ has to be a positive definite matrix for all $\underline{n}$.

Now, for the Newton's method to work; however, one point that should still be noted is that again Newton's method is also not a very guaranteed convergence because, again you are taking second order terms and you are neglecting anything beyond second order. So, naturally this also imposes some condition and the condition that is imposed is that this Hn matrix has to be a positive definite matrix for all n, now this is something should be noted that when I say that Hn has to be positive definite matrix for all n

Then, it is not really possible for us to guarantee that, it will be positive definite for all the iterations, may be that we start with again Hessian matrix that is positive definite, so H 0 we can control, because H 0 is our initial guess, we may be having a control on that, but we never know that ultimately what Hn is going to be. Because Hn is very much dependent upon the iteration that progresses, so there cannot be any guarantee that Hn will be positive definite for every iteration.

So, generally again this is true that in most of the cases, we have that asymptotically convergence is obtained for that, now in the next class, we will be discussing about another methodology for the unconstrained optimization. And before we go into that in fact, we will be dealing will be spending of bit of time with the matrix differentiations theory, so that if you are having any doubts about the differentiation of this expressions, those doubts can be solved.

Thank you very much.