**Intelligent Systems and Control**

**Prof. Laxmidhar Behera**

**Department of Electrical Engineering**

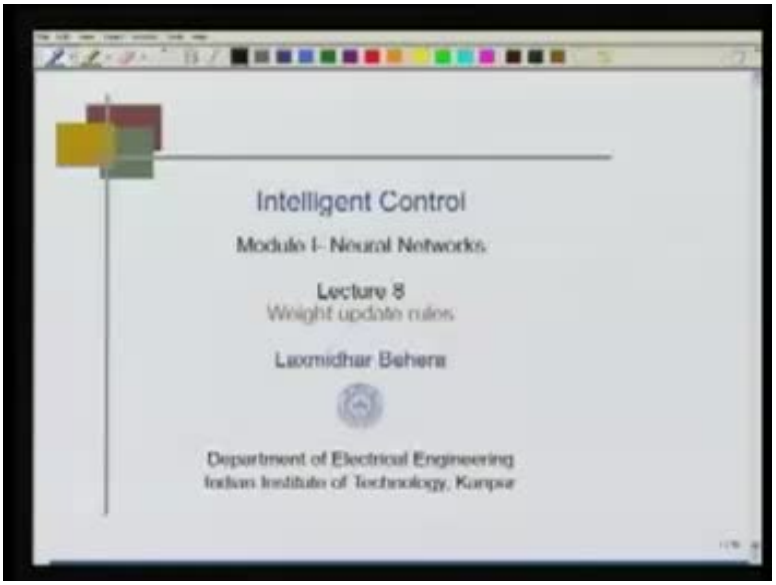
**Indian Institute of Technology, Kanpur**

**Module No. # 01**

**Lecture No. # 08**

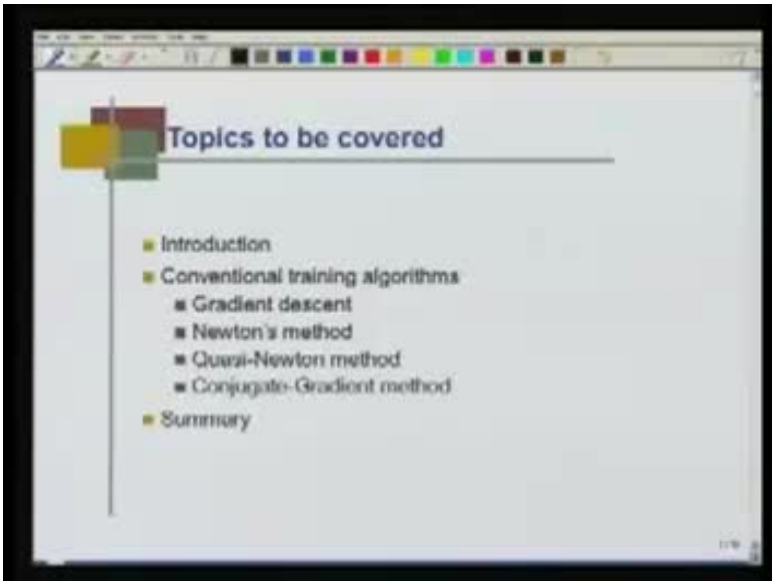
**Weight Update Rules**

(Refer Slide Time: 00:29)



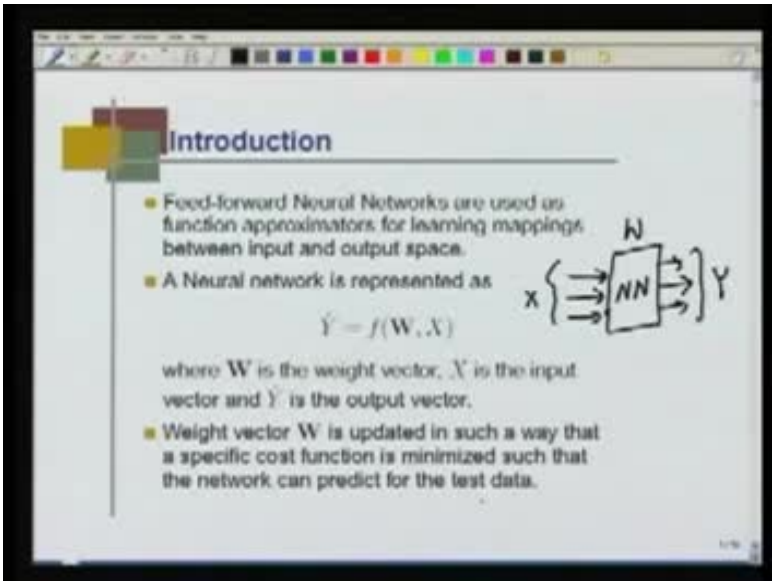

Welcome everybody. This would be a lecture on weight update rules for neural networks, various possible schemes. In this course on intelligent control, we have already discussed a various feed forward neural networks, multilayer neural networks, radial basis function neural network and associated algorithms for these networks. Also it proposes in the last class, a very interesting algorithm for weight update that was the adaptive learning rate using lyapunov function. We would now, summarize the various schemes that are being employed to improve on the convergence property, as well as the global convergence criteria for minimizing a cost function given a network.

(Refer Slide Time: 01:50)



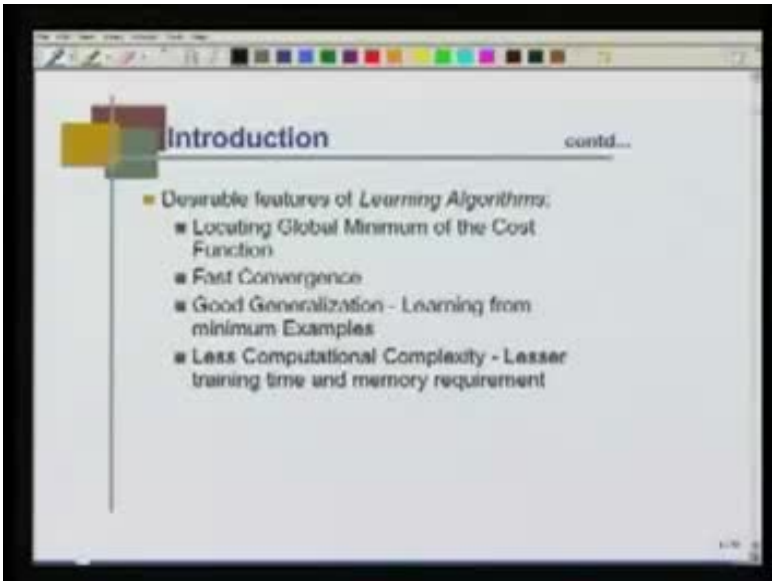

So, it would be a kind of a summary to let you know, what is the state of point of (( )) various weight update rules for specifically feed forward networks. Topics that would be covered today would be conventional training algorithm, gradient descent that we have already being talking about, which is popularly known as back propagation algorithm for feed forward networks, Newton's method quasi newton method, conjugate gradient method. And these methods would be kind of compared with our schemes also that we propose last time that is our personal work. The work of our group at I I T Kanpur on adaptive learning rate and then we would kind of summarize everything that we have learnt.

(Refer Slide Time: 02:42)



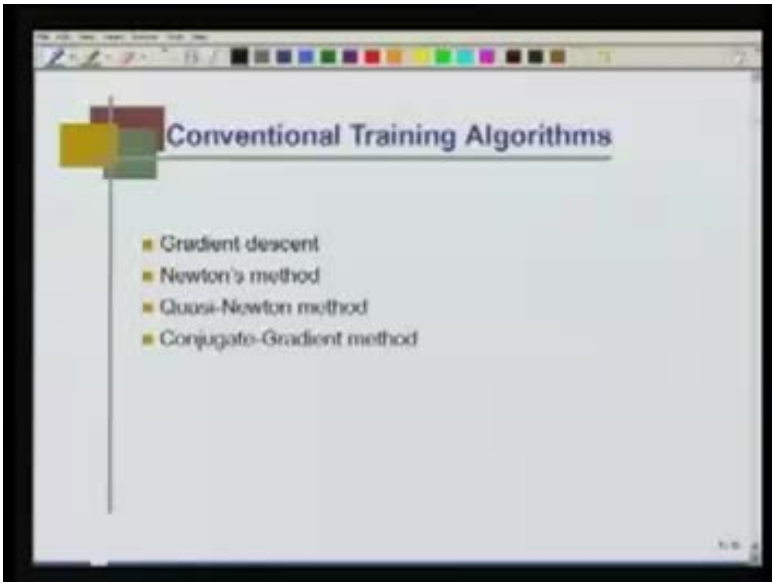

So, introduction feed forward neural networks are used as function approximators for learning mappings between input and output space. A neural network is represented as Y equal to f W X that is, if I have a neural network here, this is some neural network then my output vector is Y and input vector is X neural network is characterized by weight vector W then my output of neural network is a non-linear function of weight vector W and input X, weight vector W is updated in such a way that is specific cost function is minimized such that network can predict for the test data.

(Refer Slide Time: 04:02)



So, we give some data, we generate some training examples for this network for training and then with another set of such examples, we taste the predictability of this network. Desirable features of learning algorithm: locating global minimum of the cost function; fast convergence; good generalization that is learning from minimum examples. So, I take minimum examples, and then this network is provided with minimum example and from that example, it can if the network in extract the map of the entire data that is there in input-output space for that specific domain and the prediction is pretty nice, then that is a good generalization, less computational complexity. So, this is an important, we do not want to have an algorithm that takes a long time for computation to a very fast like, back propagation algorithm with all its drawbacks is very fast simply a first order approximation of taylor series, approximation the back propagation that tries to minimize. So, it is very fast this algorithm is…

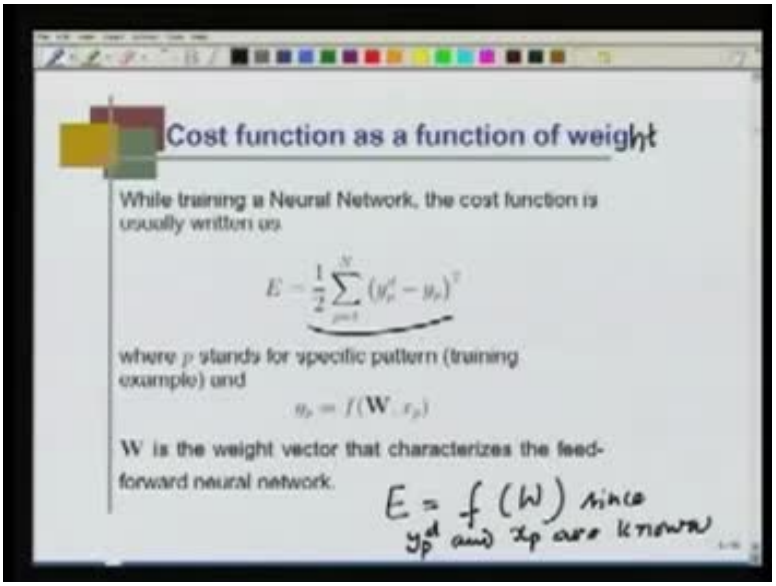
(Refer Slide Time: 05:28)




So, the traditionally that people have tried to propose various algorithms for feed forward neural networks including the even feedback neural networks that we have already talked all these things can be applied both to feedback and feed forward network for system identification, because we are talking about intelligent control. So, when we talk about neural network, we are talking in terms of either system identification or control.

So, all these gradient descent Newton's method, quasi newton method, conjugate gradient method, all these methods the variant of these methods have been applied to neural network to varying degree of success.

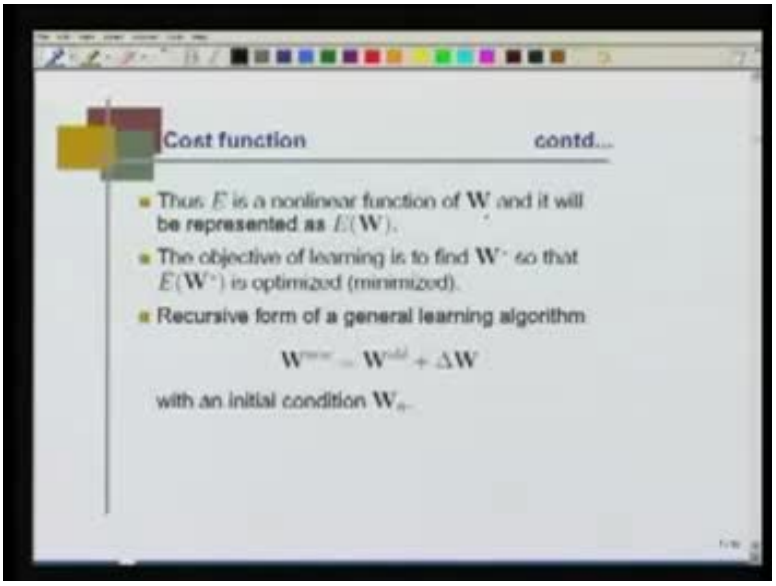
(Refer Slide Time: 06:06)




So, normally a cost function is a function of a weight, you see that this is… so, while training a neural network, the cost function is usually written as this, where this is your desired output, this is your actual output predicted by the neural network and then for and p is the stands for pattern. So, if I have N patterns. So, this total error square of the errors should be minimized. Normally, this is a quality cost function but not necessarily we will go for this function but usually researches they use this kind of function this type of function.

And here, if you look at what is y p? This y p is the output of the neural network and y p is function of neural network weight and x p is the neural network input. So, this x p is input y p is output. So, in that sense you can easily see that E is actually a function of W because since y p d and x p are known.

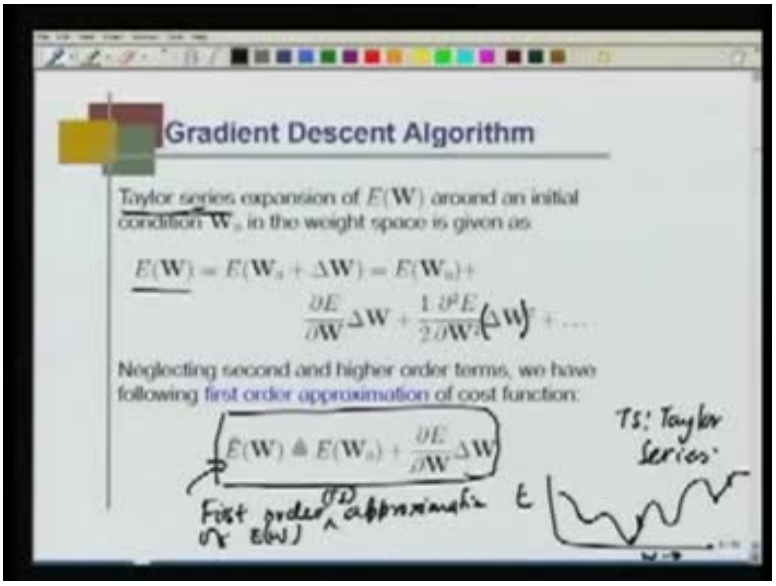
(Refer Slide Time: 08:11)




Given a training example, given a training set, then I know what is y p d in y p also this function, I know what is x p; so, all that I do not know is the weight vector. So, in that since my cost function is simply a function of weight vector. Thus, E is a non-linear function of weight vector W and it will be represented as E W, you understand why we are writing, E W means E is a exclusively a function of W, when we train in neural network.

The objective of learning is to find W star, such that E W star is optimized or minimized. Usually, we minimize here. So, recursive form of general learning algorithm is weight vector new is equal to weight vector old plus delta W with an initial condition W naught, we start with some initial condition.

(Refer Slide Time: 09:10)




Now, let us talk about gradient descent, I mean what we are trying to do in this lecture is that I would like to give a comprehensive picture of all these learning algorithm, that we are being talking about until now. So, let us talk about a gradient descent algorithm, what is actually a gradient descent, what it does?

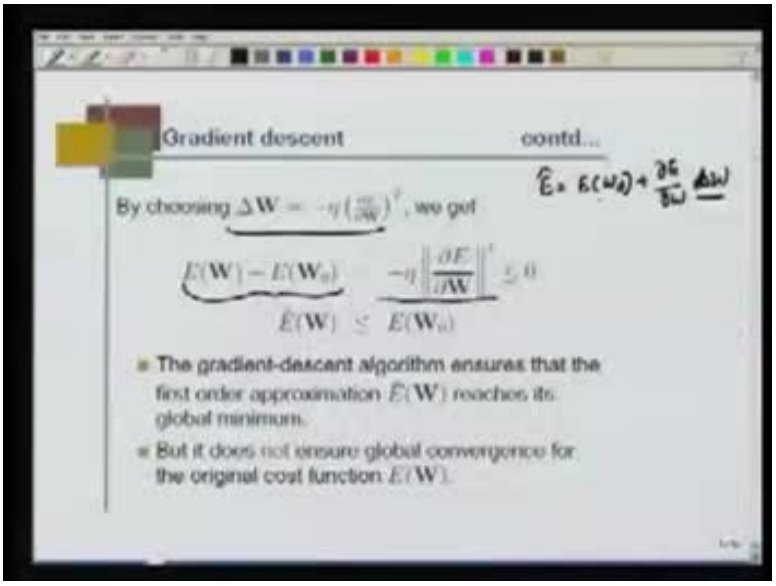
We understood in our previous discussion, it certainly does not take us to or it does not guarantee us to global convergence or theoretically the gradient descent cannot take us to to the global minimum. So, that is if we have a cost function like this E and this is W. So, obviously if I start from here I may reach here, if I start from here then I may reach here.

So, it all depends on the initial condition, where I am starting and since I do not know where is the global minimum. So, how can I fix a initial condition that will be very near to the global minimum. So, this is very difficult and no theoretical theoretically gradient descent cannot ensure global convergence it only ensures local convergence.

So, in that sense what gradient descent does? Let us think about the function, that we talked about that we are trying to minimize E W error a function. So, if I expand this error function around W naught some initial point. So, I do at Taylor series expansion, then you see that E W naught plus delta E by del W delta W first order, second order term that is del square E by del W square delta W square. This is whole square actually and plus 1.

So, this is the series, what we are talking about and if we neglect in this expansion, second order and higher order terms that is if I simply consider the first order Taylor series expansion of the cost function, then this is my expression. So, the approximation first order of this is first order approximation of E W, first order Taylor series expansion T s, T s is Taylor series. So, this is what we have written here Taylor series. So, this Taylor series expansion of the cost function, the first order Taylor series looks like this which is given in this block.

(Refer Slide Time: 12:28)



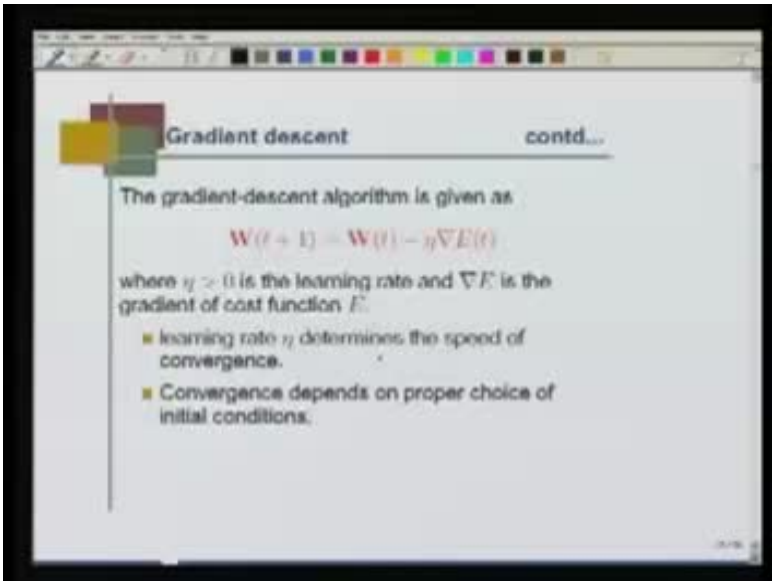

Now, if I select delta W as minus eta deli upon del W transpose E is a scalar, when I differentiate a scalar with respect to a vector it becomes a row vector. So, transpose makes it a column vector and delta W is a column vector. So, we get, if I replace this delta W. This particular expression in the previous expression and that expression is that we wrote earlier E hat is E W naught plus del E upon del W in delta W.

So, in this replace, this delta W by this minus eta del E upon del W transpose, if you do that than E minus E W will become this quantity because if you take this one here so you get minus eta deli upon del W norm square and since this is a known quantity, which is always a positive quantity. So, hence this particular term is always negative, what it implies it implies that if I start from W naught the next W the update would be always less than E W naught.

So, that is again, when this W will become the initial point and you go to the next point, so, it will always decrease - this will be always decrease because it will always decrease in which sense the decrease is ensured only when in terms of the when E W is looked at as a first order approximation of Taylor series, but this is not true I cannot say E W is less than equal to E W naught, this is E hat W and E hat W is a first order Taylor series approximation of the cost function.

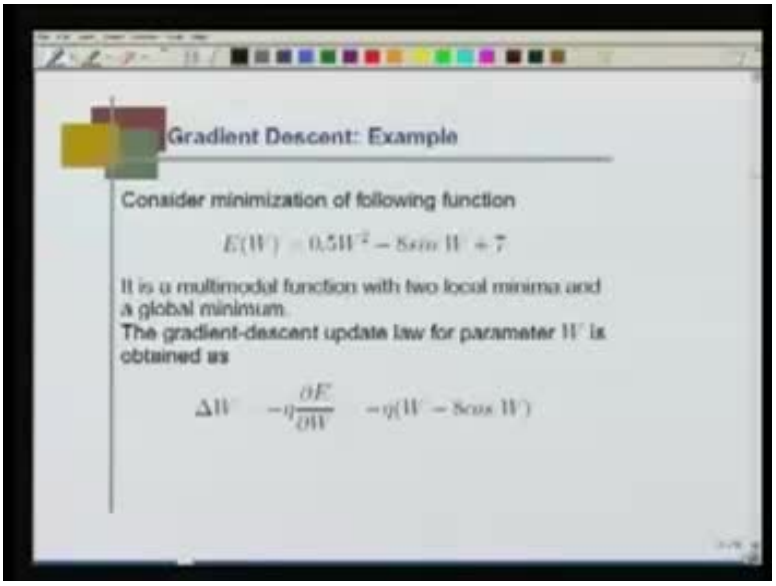
So, the gradient descent algorithm ensures that the first order approximation E hat W reaches its global minimum, but it does not ensure global convergence for the original cost function E W. So, in that sense what a gradient descent does, it tries to optimize the first order Taylor series expansion of first order, Taylor series approximation of the original cost function. So, in that sense we only reach a local convergence.

(Refer Slide Time: 15:46)



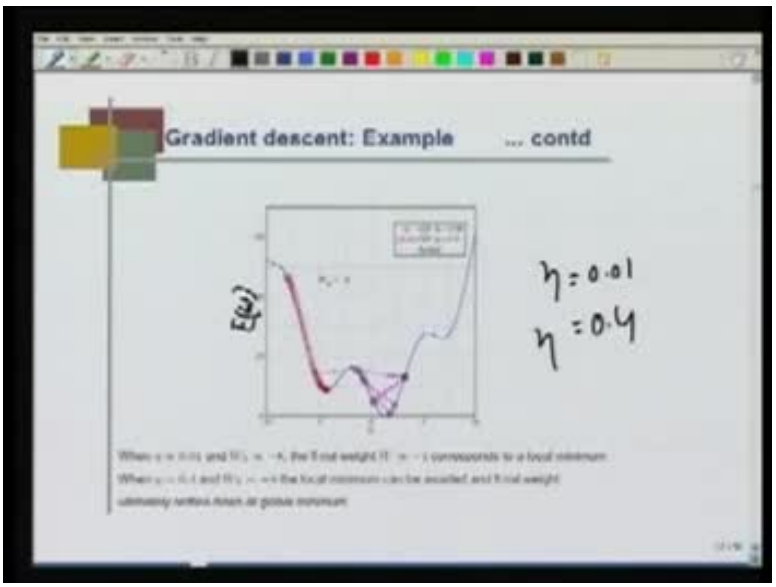

So, the gradient descent algorithm as usual is given by this. So, learning rate eta determines the speed of convergence. Convergence depends on proper choice of initial condition that I have already told you.

(Refer Slide Time: 16:01)




So, now let us take a simple example of a gradient descent: E W is 0.5 W square minus 8 sin W plus 7. So, this is a non-linear cost function, so it is a multimodal function with two local minimum and a global minimum.

(Refer Slide Time: 16:39)

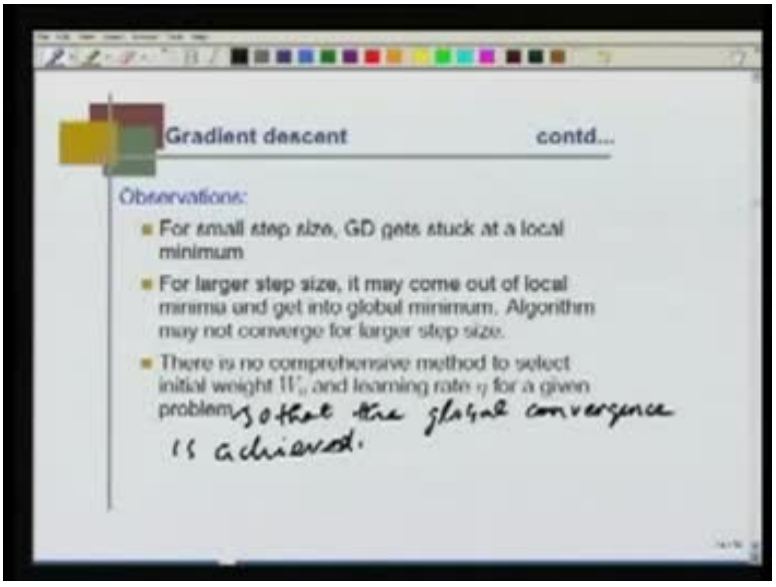



The gradient descent update law for parameter W is obtained as this. So, delta W is minus eta del E by del W. So, if I differentiate this I get eta, this is 2 into 0.51. So, W minus 8 cos W and this is 0, so you see that this particular cost function, this is actually e w. So, this E W it has two local minimum and one global minimum. So, interestingly, if

you look at the result is if I use adaptive value eta equal to 0.01, which you are seeing that is the rate one circle one. So, if I start from this point w not is minus eight, if I start from there. So, what I am seeing that this comes and comes slowly converges at this point and does not go from there. So, it actually converges to local minimum. So, if I start from here I reach here but if I change eta equal to from 0.01 to eta equal to 0.4, if I take, then my eta size, step size learning rate size has increased a little bit. So, you can easily see that is square one. So, from this point it jumps to this point and from this point it goes to this point and again from here, until it converges to global minimum in this case.

So, changing this eta, I am able to again reach in this case to the global minimum. So, when eta is 0.01 W naught is minus 8, the final weight W equal to minus 4 corresponds to local minimum this one minus 4. When eta equal to 0.4 and W naught is minus 8, the local minimum can be avoided and final weight ultimately settles down at global minimum at this point. So, I reach the global minimum but this is simple function and because this is a simple function we are able to show that by changing this eta intelligently.

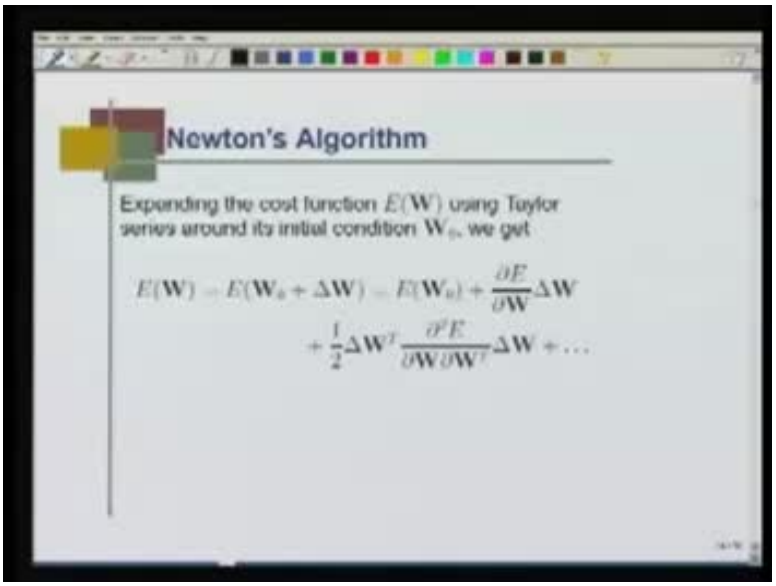
(Refer Slide Time: 18:58)



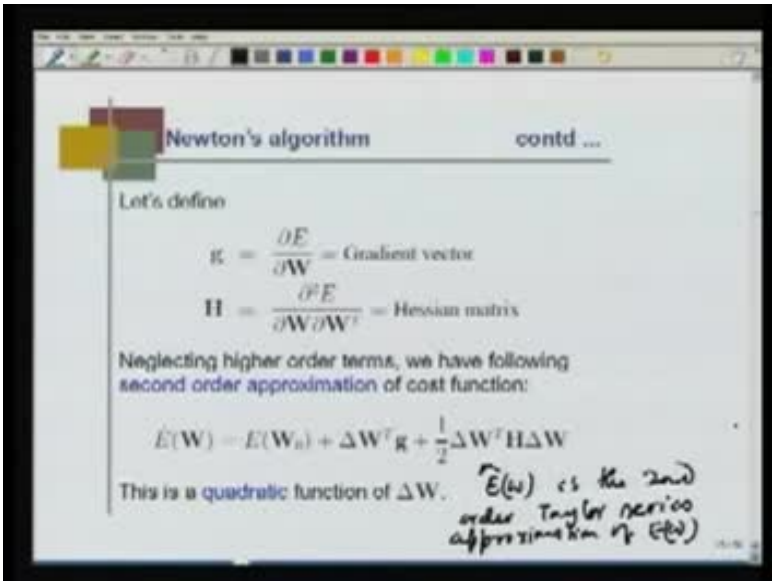

We can reach it, but if the function is complex, we have no idea, we have no comprehensive method to reach the global minimum using gradient descent; all though heuristically we can reach, you know like near global minimum. So, observation for

small step size gradient descent gets stuck at local minimum for larger step size it may come out of local minimum and get into global minimum algorithm may not converge for larger step size as well. There is no comprehensive method to select initial weight W naught and the learning rate eta for a given problem. So, that the global convergence is achieved. So, that the global convergence is achieved. So, that is about gradient descent now, because gradient descent is very slow in its convergence it does not guarantee global convergence; so people thought about Newton's algorithm where like in gradient descent the weight update law is such that it tries to find the global minimum of the first order Taylor series approximation of the cost function.

(Refer Slide Time: 20:22)

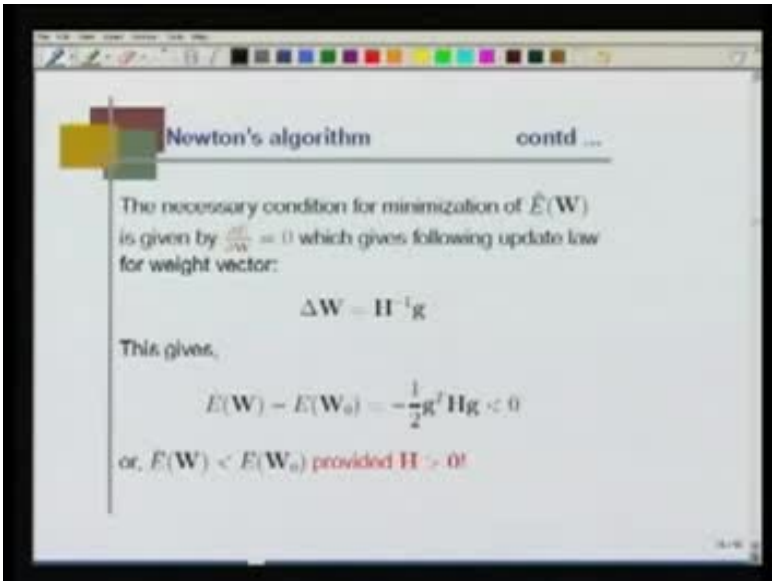


(Refer Slide Time: 20:31)

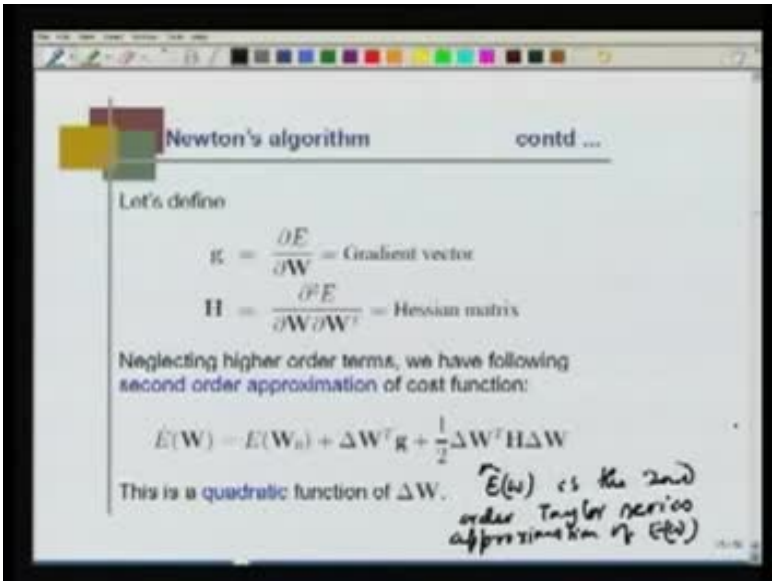



Now in Newton algorithm, if we take the same cost function e w and again we expand its Taylor series and we take the second order of approximation here. So, if we take second order approximation. So, here, let us define g as del E upon del W the gradient vector and there is a Hessian matrix that is defined as H, which is does square E upon doe W doe W transpose. So, neglecting higher order terms, we have following second order approximation of the cost function. So, this is the second order approximation; here E W is the second order Taylor series approximation of actual cost function E W and that we write this particular form. So, the all other higher order terms from this point onwards has been neglected.

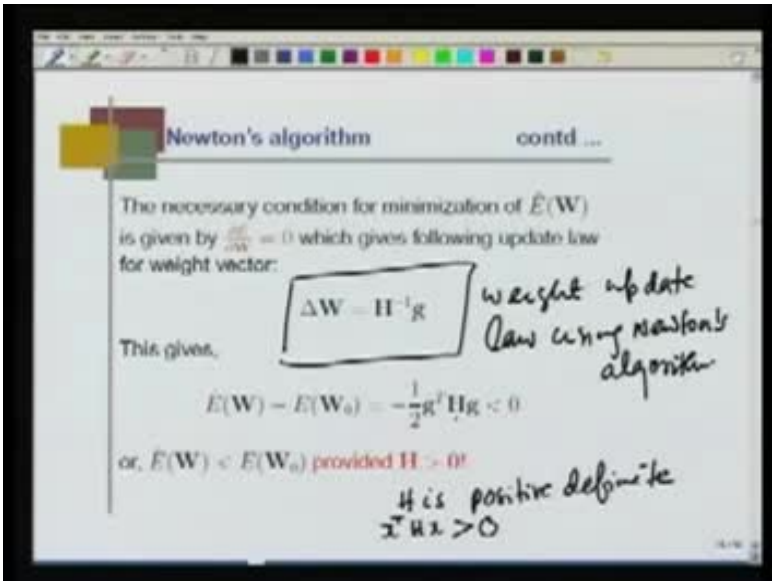
(Refer Slide Time: 21:38)




(Refer Slide Time: 20:31)



(Refer Slide Time: 22:21)

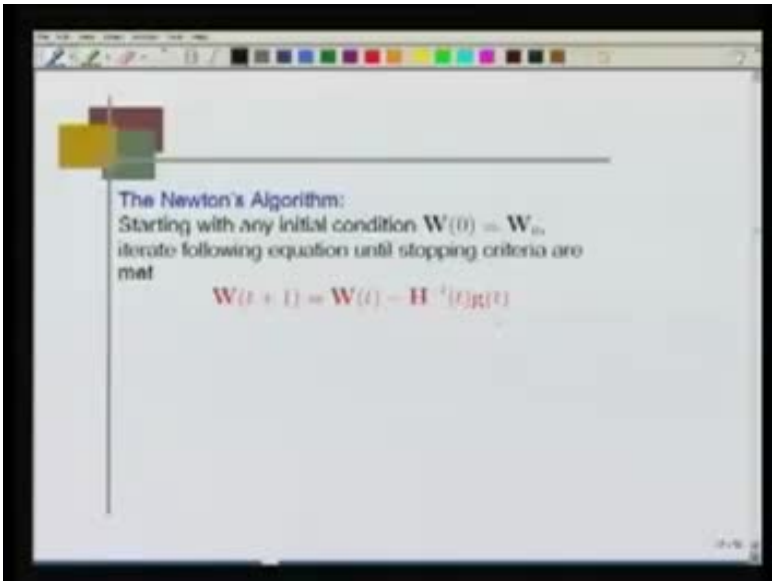



So, this is a quadratic function of delta W. The necessary condition for minimization of E hat W, which is the second order approximation of actual cost function is given by del E upon del W is 0, which gives the following update law for weight vector, which delta W H inverse g, so that is if I go here and if I differentiate this del E W by del W this is a constant term goes away. So, differentiating this I get delta W is H inverse g; this is my weight update law using Newton's algorithm.

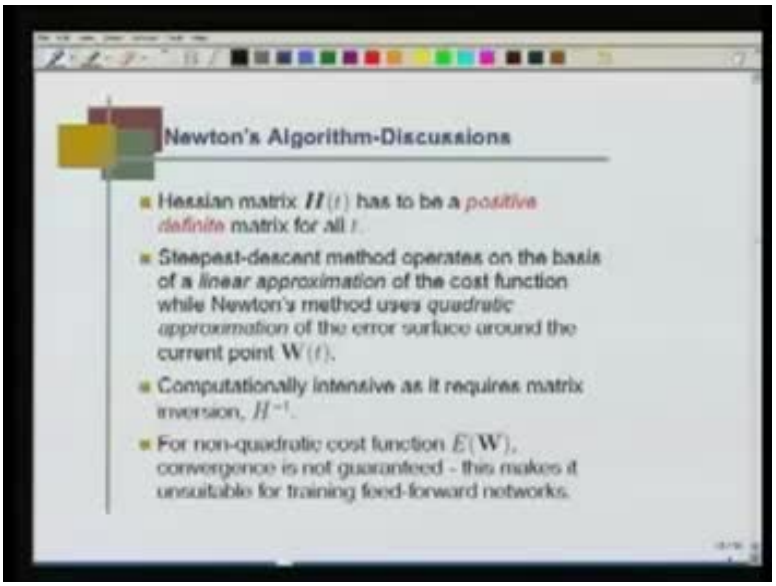
But unfortunately, of course now once I take this, you can easily see that E hat minus e w not is always a negative definite quantity provided not always it is negative definite only when H is positive definite; so this is true provided H is positive definite means H is positive definite. I hope all of you understand positive definite weight, if I take any vector extras force H x, if H is a positive definite matrix, this quantity is always a positive quantity for any accept, any in all vector you take any vector x this will be a positive quantity so that is H is positive definite. From that sense that this algorithm is not a full proof algorithm, because even it does not guarantee a locally stable unless H is positive definite - that is the biggest drawback of a Newton algorithm, but still researches have used it because it is very fast, but you see that it requires this weight update algorithm requires H inverse; inverse of a Hessian matrix imagine a normal network has at least 100 of weights. So, W and the minimal order is 100.

So, H has at least 100 by 100. The dimension of H would be at least 100 by 100, we have said not100 by 100, because it is a double derivative. So, it dimension is huge and inversion is computationally very expensive. So, in that since this algorithm all though we can use this algorithm to solve a simple cost function optimization, but in terms of neural network this is a I would say it is a infeasible algorithm, we cannot implement it because Hessian cannot be computed, it will take lot of time.

(Refer Slide Time: 25:15)



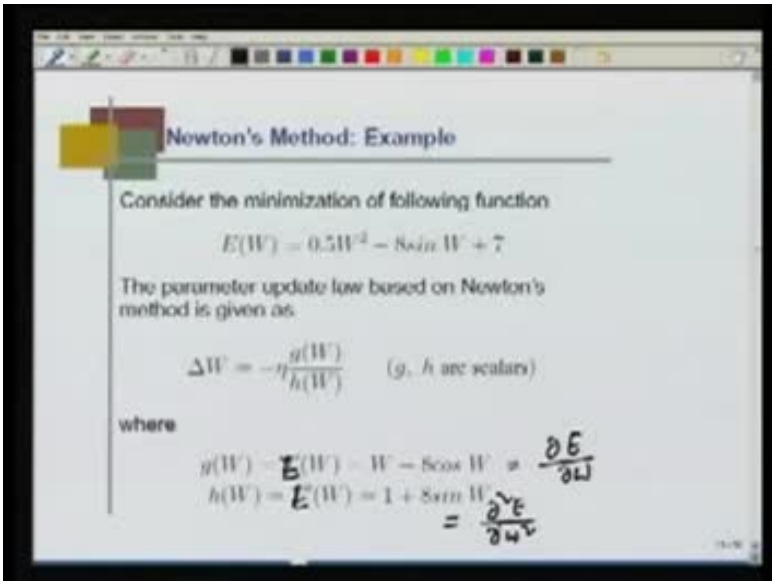

(Refer Slide Time: 25:32)



So, the Newton algorithm starting with any initial condition W naught iterate following equation until stopping criteria is met, which is W t plus 1 is W t minus H inverse g t. This is a Newton algorithm and the discussion is that the Hessian matrix H t has to be positive definite matrix for all t that has to be you cannot guarantee H all the time.

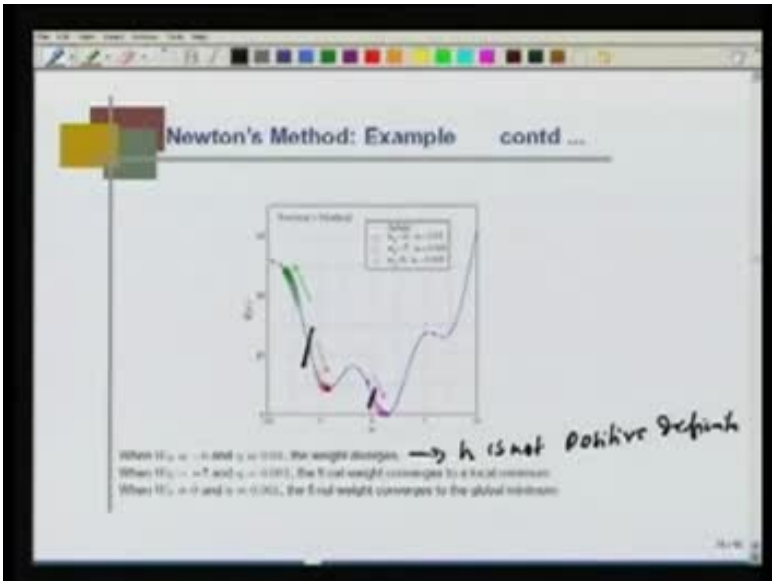
Steepest descent method operates on the basis of a linear approximation of the cost function, while Newton's method uses quadratic approximation of the error surface around the current point W t computationally intensive, whereas it requires matrix inverse in H inverse for non-quadratic cost function E W convergence is not guaranteed this makes it unsuitable for training feed forward network.

(Refer Slide Time: 26:10)




Now, we will take the same example that we consider earlier, E W is a scalar, we see that W is a scalar it's no W is not a vector here simply a scalar. This is a very simple function 0.5 W square minus 8 sin W plus 7 and if I use the because W is a scalar naturally del E upon del W is a scalar which is g and H is del E by del square E by del W square. So, that is also a scalar and weight update law delta W is minus eta H inverse; so H here is a scalar so its inverse into g that is our algorithm and where g W is f dash W, you know see this is not f this is E E dash W this is E double dash that is this dash represent this is actually del E by del W and this is del square E by del double square.

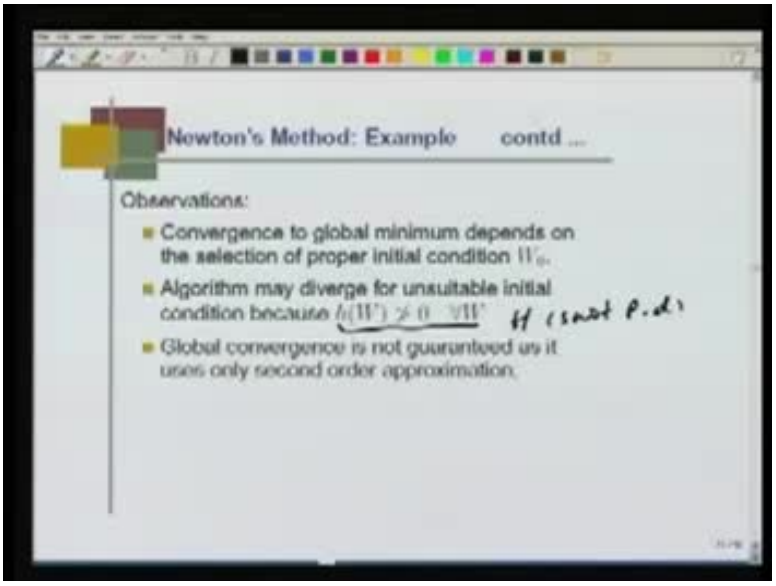
(Refer Slide Time: 27:48)




So, this represent this, and this represent this and using this, if you update, you see that for some initial condition, when W naught is minus 6 it has 0.01. The weight diverges that is because H is not positive definite so in this case, H is not positive definite; so H is a scalar here it is obviously it is not greater than 0, it is a negative quantity and when W naught is minus 7 and eta is 0.001. The final weight converges to local minimum here and when W naught equal to 0, so if you start from here you reach here, but if you start from here you reach here, the final weight converges to the global minimum.

But then we really did not achieve much with respect to in compresentive gradient descent except, probably the gradient descent even converging to the local minimum. Whatever the number of steps it will take to go to the local minimum, probably using this Newton method, you can improve on that speed but no way the desired goal of learning is achieved by Newton method.

(Refer Slide Time: 29:17)




So, observation are as follows convergence to global minimum depends on the selection of proper initial condition W naught. Algorithm may diverge for unsuitable initial condition because H W is not a positive quantity or it is in fact in general I would not say H is a scalar for our example, but in general H the Hessian matrix is not positive definite or is not positive definite always.