Neural Network and Applications

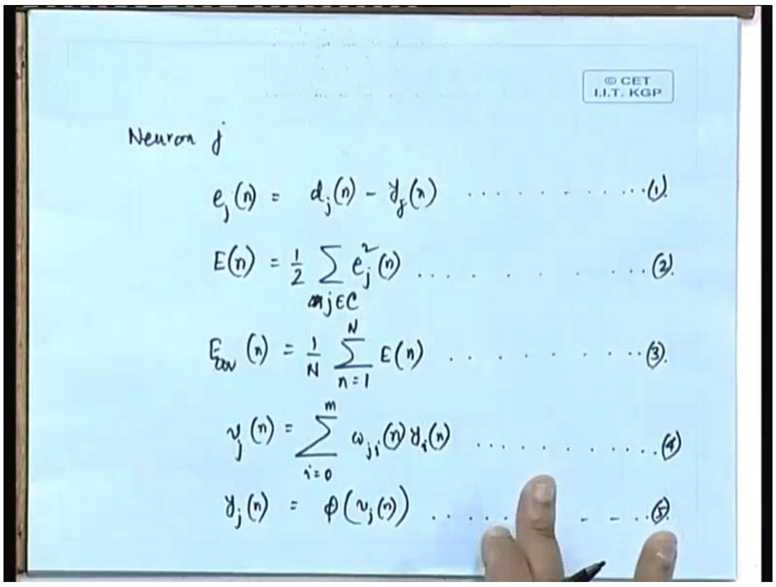
Prof. S. Sengupta
Department of Electronic and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 19
Back - Propagation Algorithm

Today's lecture is Back Propagation Algorithm, this is something which I wanted to introduce in the last class. Again towards the end of last class, we had just started giving the ideas about the back propagation, because major part of the last class was actually spent in discussing the concluding part, related to the single perceptron.

And then we had introduced the multi-layer perceptron, that basic algorithm in which the multi-layer perceptron's operate is the back propagation algorithm. So, I mean whatever we had introduced in the last class, let me just write down those equations without much of explanation, because I think that those are pretty simple. So, what we took was the j th neuron take any neuron of the system. And let us say that we take the output neuron only, we take any one of the neurons from the output layer.

(Refer Slide Time: 02:17)



Neuron j

$$e_j(n) = d_j(n) - y_j(n) \quad \ldots\ldots (1)$$

$$E(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) \quad \ldots\ldots (2)$$

$$E_{av}(n) = \frac{1}{N}\sum_{n=1}^{N} E(n) \quad \ldots\ldots (3)$$

$$v_j(n) = \sum_{i=0}^{m} \omega_{ji}(n)\, y_i(n) \quad \ldots\ldots (4)$$

$$y_j(n) = \phi(v_j(n)) \quad \ldots\ldots (5)$$

And let us say that we take the neuron j and at the iteration number n. So, what we had written down are the arrow term that, is e of j at the iteration n is equal to d j n minus y j n where, d j n is the desired output at the jth neuron y j n is the actual output and this let us call it as equation number 1. Then, we had defined the instantaneous value of the error energy E n error energy at the alternation n to be equal to half of summation e j square s n and we varied n from 1 to how neurons there yes I mean this is j indexed as j.

So, j belonging to the set c which includes all the neurons in the output layer, so this was our equation number 2. And then, we had defined the average square error energy E average n to be 1 upon N summation n equal to one to capital N, E n where, what is capital n, capital N is the number of patterns that we are presenting, total number of iterations.

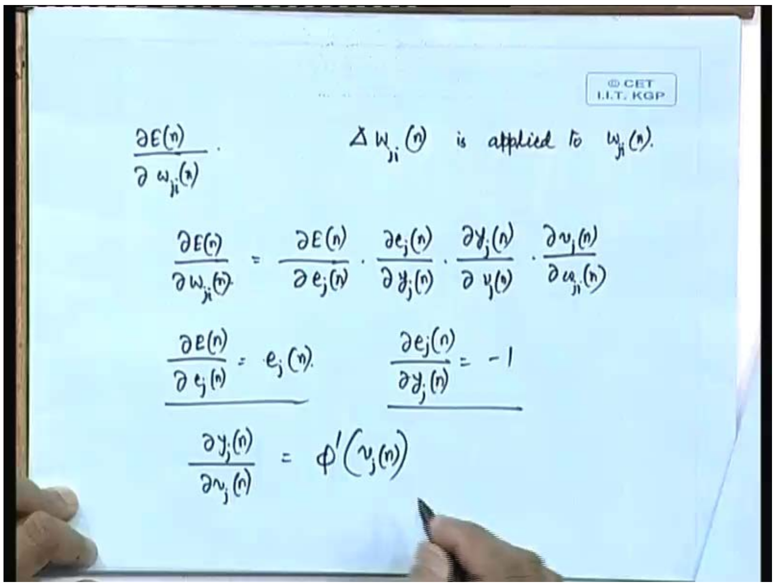
Because, what we are doing is that with every pattern presentation, we have been calculating this e j's. And consequently the E n's will be calculated as the summation of all this square terms and e average will be the average over all these E n's all these n's. So, this we call as equation number 3, which is the average square energy, now we can write down the induced local field at the input of the neuron j.

So, we are taking the neuron j and at the input of the neuron j if we are taking the activation function. The input to the activation function, because what we are doing is that, we are summing up all the weighted summations we are calculating and those weighted summations will be actually will be modified by the phi function. So, phi as an argument of v j, so where v j is called as the induced local field.

So, we are calling v j as the induced local field where, v j of n we can write down as what as the summation of w j i, w j i at the iteration n times y i n where, y i belongs to which layer, the layer which is previous with yes previous weight. So, if we are taken the output layer j as any output layer neuron, then; obviously, it refers to the hidden layer, that just precedes the output layer. And we have to sum it up for i is equal to 0 to m, where m is the number of neurons in the previous layers.

So, this we are calling as equation number 4 and the actual output y j of n, that is to be written as what the phi function of activation function of v j that is correct and this is equation number 5. So, these are the five basic equations with which we begin our analysis. In fact, what follows is a very straight forward mathematical treatment.

(Refer Slide Time: 06:22)



© CET
I.I.T. KGP

$\frac{\partial E(n)}{\partial w_{ji}(n)}$. $\Delta w_{ji}(n)$ is applied to $w_{ji}(n)$.

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

$$\underline{\frac{\partial E(n)}{\partial e_j(n)} = e_j(n).} \qquad \underline{\frac{\partial e_j(n)}{\partial y_j(n)} = -1}$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \phi'(v_j(n))$$

And we are going to calculate the partial derivative of dou E n dou w j i with respect to n. Why, because the change of weight that is delta w j i that we are applying to the synoptic weight, so delta w j i n is applied to the current weight that is w j i n. And this delta w j i that we are going to get is proportional to this dou E n dou w j i n. So, this we can write down as dou E n dou w j i n if I have to calculate.

Let us again go back to this expression, let us understand that where are we going to calculate this ((Refer Time: 07:14)). If we are getting this, so we will have to apply it to the equation 2 very directly, but in equation to be find that everything is written in terms of e j square. Whereas, we would like to get the derivative in terms of w j i, so ((Refer Time: 07:35)) bit of a problem.

Actually, we could have substituted I mean what we have to do is to substitute e j n from this equation number 1, where it is to be expressed in terms of v j and y j. And again y j will be expressed as the v j and v j is expressed as w j i and then only we can integrate and then only we can differentiate with respect to w j i. So, little bit of cumbersome if we try to substitute this form of v j and then substitute everything using this equation, it would be mathematically very much cumbersome.

So, instead of doing that what we will be doing is just we apply the chain rule of differentiation. Simply, because here in the equation number 2 n is written in terms of e j of n all right, the best thing will be is that, if we first calculate dou E n with respect to dou E j. That, we can very simply calculate from equation number 2 in which case we have to multiply this partial derivative by dou E j doe w j i n, again note that there is no direct relationship between dou E j and dou w j i.

So, we can only calculate dou E j dou y j, so if we write down this term dou E j dou y j as the second expression in the chain rule of differentiation. Then, we can calculate that very easily in fact, it is going to be minus 1 simply. In that, case we have to follow it up with a dou y j and that expression we have to have with respect v j, where we will be getting the term as phi dash v j of n where, phi dash is the derivative of v j of n with respect to derivative of phi v j n with respect to v j n. And then, again we have to take the last term as dou v j n with respect to dou w i j n, so that will complete my chain rule of differentiation. So, we have to write ultimately this thing as dou E n dou E j that is what we agreed for the first term in the chain rule. The second product term in the chain rule should be dou E j n with respect to dou y j n.

The third term should be dou y j n with respect dou v j n and the last term in the chain should be dou v j n dou w j i yes. So, this simple expression is being written as a product of four terms, but getting the product of this four terms is very easy, because what we can simply get is dou E n dou E j of n, we get very easily what is that dou E n dou e j n.

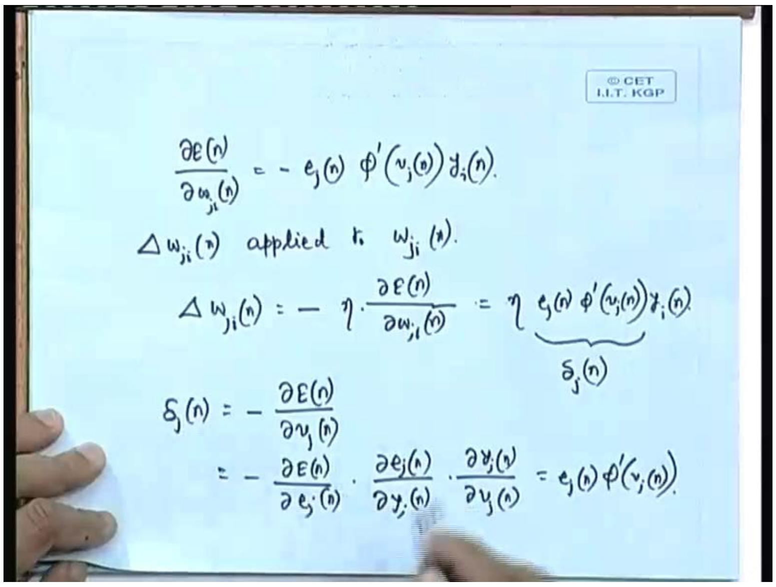
Student: ((Refer Time: 10:42))

It is simply equal to e j yes e j n. So, this is equal to e j of n very correct and what is dou E j n dou y j n that is very correct simple dou E j dou y j n. So, simply it take the derivative d j is not varying with respect to y j, so dou E j dou y j is simply to be equal to minus 1. Then, what other terms are required, so, this term is computed, second term is computed, the third term is dou y j n dou v j n and what is that, that is simply equal to phi dash we can exactly compute it only when we know the activation function.

So, typically we need a activation function, in order to exact the calculate this. But, in general given any phi we can express it a in terms of phi dash, where phi dash is the partial derivative of the phi function with respect to the activation input that is v j. So, this is also expressed and the last term that we are having is dou v j n dou w j i n and what is that, that is simply equal to y i of n.

This is simply using the equation number 4 followed. So, these are the four expressions that we compute all right. So; that means, to say that now this dou E n dou w j i n is expressible in a very manageable for this term is very nice minus 1 this term is E j. So, it is minus E j phi dash v j n y i n simply dou E n dou w j i.

(Refer Slide Time: 12:50)



© CET I.I.T. KGP

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = - e_j(n)\, \phi'(v_j(n))\, y_i(n).$$

$\Delta w_{ji}(n)$ applied to $w_{ji}(n)$.

$$\Delta w_{ji}(n) = - \eta \cdot \frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \eta\, \underbrace{e_j(n)\, \phi'(v_j(n))}_{\delta_j(n)}\, y_i(n)$$

$$\delta_j(n) = - \frac{\partial \mathcal{E}(n)}{\partial v_j(n)}$$

$$= - \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\, \phi'(v_j(n)).$$

So, we can express it as follows doe E n dou w j i n is equal to minus e j n phi dash v j n y i n. And now, the correction delta w j i n which is applied to w j i n that can be written as, because delta w j i n is going to be proportional to dou E dou w j i is not it error with respect to j i. So, what is that constant of proportionality.

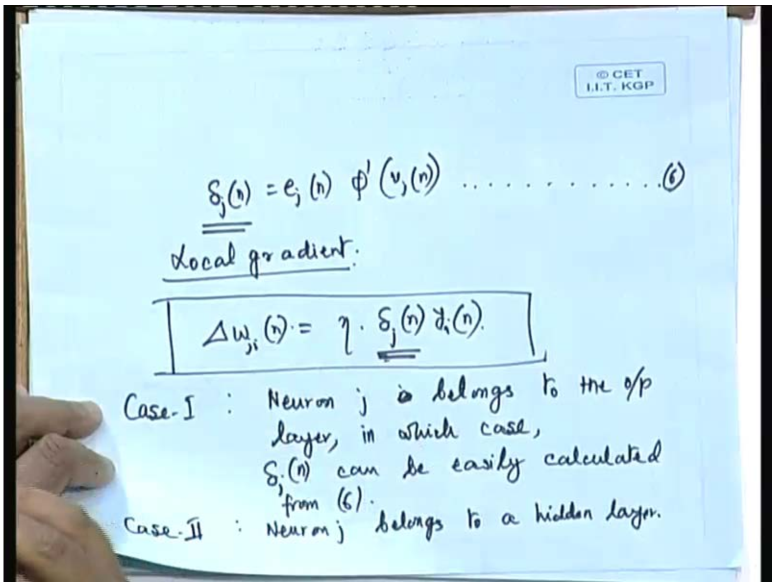
Student: eta.

Eta, so it is equal to minus eta of dou E n dou w j i n, because it should be in the direction negative to the direction of the gradient. So; that means, to say that it is simply becoming equal to eta product this term eta product e j n phi dash v j n times y i n. And now, what we are going to call this term, you see this term is nothing but, the error multiplied by the derivative of the activation function. This term we are expressing we are substituting another term delta of j, because everything is here index with respect to j.

So, delta j at iteration n replaces this product term. So, we define delta j this way actually delta j we originally define like this, that delta j n will be defined as the derivative of this error energy, that is dou E n with respect to dou v j of n. That means, to say what, that the derivative of the error with respect to the activation input which is nothing but, the which is this term dou j that we are going to delta j that we are going to write.

So, this in fact, could be expressed as minus dou E and dou v j we could express it as dou E n dou e j n times dou E j n with respect dou y j n with respect then dou y j n dou v j n and this whole thing ultimately boils down to nothing but, e j n phi dash v j n. So, this is an important expression, so this is delta j I will write down in the next page.

(Refer Slide Time: 16:05)



© CET I.I.T. KGP

$$\delta_j(n) = e_j(n)\,\phi'(v_j(n)) \quad \ldots\ldots\ldots\ldots (6)$$

Local gradient.

$$\boxed{\Delta w_{ji}(n) = \eta \cdot \delta_j(n)\, y_i(n).}$$

Case-I : Neuron j is belongs to the o/p layer, in which case, $\delta_j(n)$ can be easily calculated from (6).

Case-II : Neuron j belongs to a hidden layer.

So, this will be equal to delta j n equal to e j n phi dash v j n. So, this is our equation number I should call it as 6, because we already gone up to 5, so this is going to be our equation number 6. So, what is the significance of this delta j term, what is the physical significance of this delta j term ((Refer Time: 16:34)). This delta j term is nothing but, the derivative of the error with respect to v j, that is to say it is own induced field v j is it is own induced field.

And this is nothing but, the gradient and this is the gradient with respect to the induced field and this is a gradient that is very much local to that of j. So, we are going to call this term that is delta j as the local gradient, so with respect to the local gradient, we can express the change of weight that takes place. That is, delta w j i n is equal to eta the learning rate times delta j n, that is the local gradient times y i n, which is the input.

So, you see this is a very interesting equation that we have got, that is delta w j i n that change of weight equal to the learning rate times the local gradient times the input to the neuron j, which is in the form y i n. So, if the since y i n will be known eta is a given learning rate, what we have to get is the local gradient. Now, the most important thing is that how to compute this local gradient.

Now, local gradient computation is very easy ((Refer Time: 18:15)) as long as we have the neuron j as the output neuron, actually our analysis began with that we began with the neuron j to be the output neuron. And given that neuron j is the output neuron, this is what we get as the delta j, that is error times derivative of the activation function. Which we know error is easy to calculate, because we will be feeding the desired we will be feeding the training pattern where we will be giving the desired output d j of n will be known to us, we will be obtaining the y j n from the network.

So, calculating E j n is not a problem and we know that what kind of activation function are we using, so we just have to take the derivative of that activation function. And then, we can calculate the local gradient, so calculating the local gradient for the output neuron is not a problem, it is quite easy.

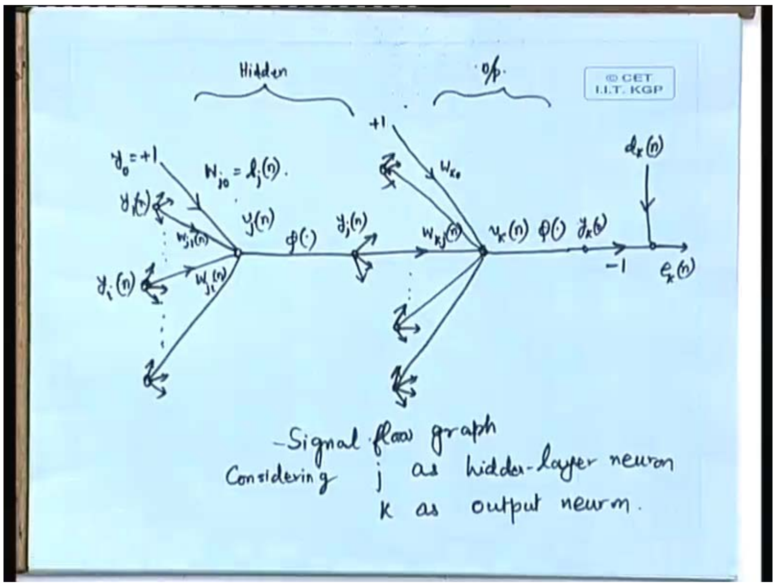
But, now if you ask me that how to calculate such kind of a local gradient for the hidden neuron. That is, why we have to do some more extra analysis, because if it is hidden neuron, then you do not know E j n do you, you do not know it because E j n you are not supplied with the desired output for any hidden neuron. Because, hidden neuron is not having any desired output like thing.

So, you will have to calculate that what this E j, you will have to calculate this delta j in some other manner and that is what we are going to show in our analysis that is going to follow. Now, depending upon the location of this neuron j, so two cases can arise, that is neuron j is the output neuron is belongs to the output layer. In which case, delta j n can be easily calculated from 3.

And case 2 we consider, where neuron j belongs to a hidden layer. And they are lies the problem, there we have to calculate something which is equivalent of this delta j in the hidden layer. So, we will have to break our analysis for the, so now, the model that we are going to consider will be as follows. Let us consider, that there is the output layer there is a hidden layer and then there is a layer that is previous to that hidden layer also.

Simplest case is, that if it is a three layer architecture, where we will be having the input layer followed by the hidden layer, followed by ultimately the output layer. So, we consider and let us just take a typical signal flow graph for that, so we consider the neuron j to be in the hidden layer. So, hidden layer means it is getting it is input from a layer previous to it.

(Refer Slide Time: 22:18)



-Signal flow graph
Considering j as hidden-layer neuron
k as output neuron.

So, let us consider that, so let us say that this is where the neuron j is there. So, here we will be having v j of n, the local induced field v j of n, which will be the summation of all these w j i times the inputs. So, here we will be having y 0 that is to say the bias which is equal to plus 1 over here and the weight, that is w j 0 is equal to b j of n which is nothing but, the bias term and then we will be having all the otherwise.

So, we will be having y 1 over here the first input and this we will be writing as w j 1 and like that we will be having some y i everything we will be writing as y 1 of n and here y i of n that is how we are going to write it. And here, the weight corresponding weight will be shown as w j i, again w j 1 at iteration n w j i at iteration n.. So, everywhere we should not be omitting this n and there will be many others to follow.

And in fact, mind you that these y's, that is to say the input that is coming from the previous layer is not only connected to this hidden neuron. But, all the other hidden layer neurons, which are there in the same layer, there will be some other neuron here, some other neuron here. So, typically we should consider the y connections like this, so this is the way whereby we indicate in the graph, that it is connected to several other neurons in the next layer.

So, all these y inputs are connected to the neurons in the succeeding layer. So, these are all our y inputs and we calculate v j n, which is the induced local fields to this neuron j, which is in the hidden layer. And then, this v j n on this v j n we will apply the activation function phi function and then we are getting what, we are getting y j n as the output, that is what we are calling y j n is equal to phi of v j n.

Now, this y j n will be connected to the output layer to all the neurons in the output layer it will be connected. But, we are considering as a specimen case we are considering a single neuron from the output layer I mean as a detail connection we will be showing that and let us call that neuron to be the kth neuron. So, that here we are going to compute v k at iteration n, where k belongs to the output layer.

So, k belongs to the output layer here and y j n will be connected to several other neurons in the output layer I mean to all the neurons which are existing in the output layer. So, that is why we are showing the connection like this and here what will be this weight, this will be w k j, where it connects it is output neuron k with the proceeding neuron j, which belongs to the hidden layer.

Student: ((Refer Time: 26:08))

Yes, very correct w k j n, everywhere we will be having n, because it is analyze with respect to the pattern number n. Now, likewise this v k n will be connected to other neurons which are there in the preceding layer. So, this is I can say that this happens to be the hidden layer and then coming to here this happens to be the output layer, so here what we are going to get is that, there will be several other inputs to this v k.

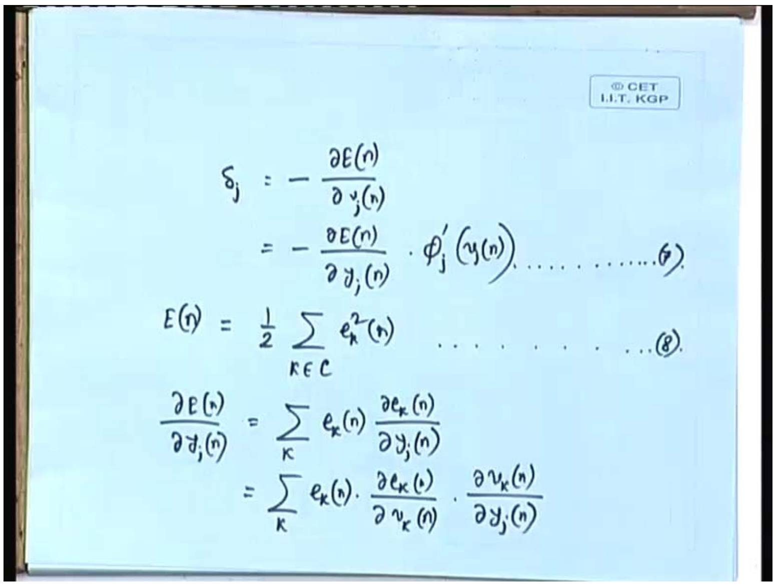
Starting with the bias that is plus 1 over here and then there will be a w k 0 n or and then there will be other inputs or other outputs of the hidden layers, which will be there. So, everything will be connected to the output layer, so like that it will get connected, so this is the way our graph would look like. So, here all this inputs are coming from the hidden layer and k belongs to the output layer.

And then, on this v k we will apply again another phi function of that and in the process we will be getting y k of n. So, y k of n will be the actual output at the neuron k in the output and likewise there will be a desired output which will be d k at iteration n. So, d k n will come here and y k n will also come with a minus sign, so d k minus y k and that will give rise to our E k of n. So, this will complete the whole signal flow graph.

So, this is signal flow graph we can say considering j as hidden layer neuron and k as output neuron. Anybody having any doubts or confusions related to this signal flow graph is it understood, let me just repeat once again in an ((Refer Time: 29:11)) we consider neuron j in the hidden layer that is connected to it is preceding layer. So, all these y's that we have shown are the outputs of the preceding layer being applied on this jth neuron.

We compute v j n, we then compute y j and from it by going to the activation function. And then, all the hidden layer outputs will be connected to the neurons in the output layer, we take a typical output layer neuron kth output layer neuron. And then we connect that to all the hidden layer neurons over here, then operate this v k with the phi function get the y k and we can compute this v k n. So, this is the complete signal flow graph that we are getting.

(Refer Slide Time: 30:15)



© CET
I.I.T. KGP

$$\delta_j = -\frac{\partial E(n)}{\partial v_j(n)}$$

$$= -\frac{\partial E(n)}{\partial y_j(n)} \cdot \varphi_j'(v_j(n)) \quad \ldots\ldots\ldots\ldots (7)$$

$$E(n) = \frac{1}{2}\sum_{k \in C} e_k^2(n) \quad \ldots\ldots\ldots\ldots (8)$$

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)}$$

$$= \sum_k e_k(n) \cdot \frac{\partial e_k(n)}{\partial v_k(n)} \cdot \frac{\partial v_k(n)}{\partial y_j(n)}$$

Now, we can compute the delta j, delta j is what delta j is going to be the local gradient for the jth neuron. Now, this time the jth neuron is the hidden layer, so delta j actually according to the very basic definition should be written as dou E n dou v j, that is the definition of the delta j, that we have been considering. So, that we instead of expressing that directly we can express it as dou E n doe y j n multiplied by dou y j n with respect to dou v j n and that is what, that is the derivative of the activation function of the jth neuron.

So, this should be phi dash j v j n is that understood. So, this is our equation number, what is the equation number running 7, we have to give equation number 7 to it. So, this is understood, so this is again by applying a simple chain rule, that instead of taking the derivative with respect to v j, we are first taking the derivative with respect to y j. And then, taking the derivative of this function with respect to v j, so that is it.

So, now we have to calculate this dou E n dou y j n, now is it very easy let see that, firstly that the En expression, what is the E n expression now half of...

Student: summation.

Summation.

Student: E j square.

E j square are you very sure about it.

Student: ((Refer Time: 32:14))

Divided how some people are saying E k square, some people are saying E j square.

Student: E k square.

E k square, because k belongs to the output layer and when it comes to the question error energy, the errors are all computed at the output. Because, that is where we can only compute the ((Refer Time: 32:35)) this output is only available out here, so whatever E k we can calculate that is only at the output layer. And we have been using j to be the hidden layer and k 2 with the index for the output layer.

So, we should write E k square n where K belongs to the set C, but C is a set containing the output neurons. So, here, so this one we are going to call as the equation number 8, so now with that idea let us try to compute dou E n dou y j n. Now, this could be calculated as follows, that we can simply write it actually this we can first differentiate with respect to E k. And then E k can be differentiated with respect to y j and that we have to sum up over all the case is not it, taking anyone E k would give you E k multiplied by dou E k dou y j.

So, we are summing it up over all the neurons in the output layer. So, this is summation of summation over E k n dou e k n dou y j n is that correct. So, this one now again dou e k dou y j is still not very easy, because k belongs to the output and k belongs to the hidden. So, we have to do some further mathematical manipulation, so let us carry on, so we can express it as summation E k n dou e k with respect to dou v k this we can calculate.

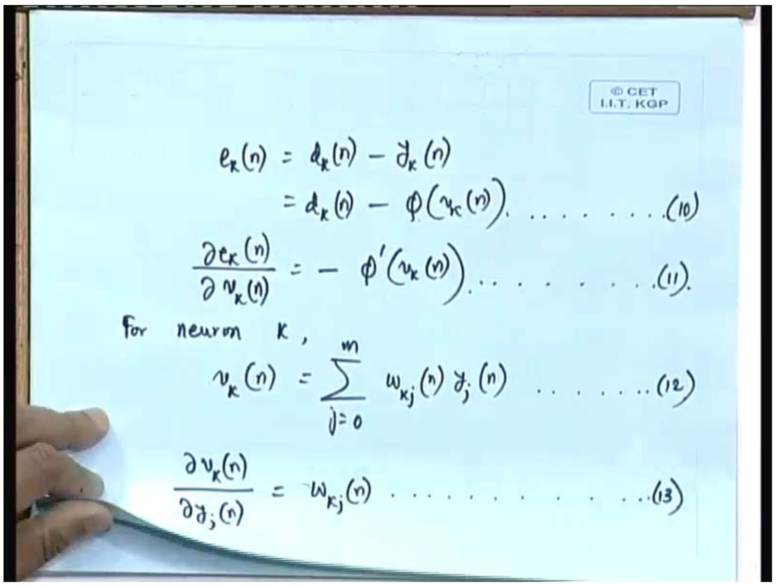
Because, everything it is pertaining to k only and then what remains is dou v k n dou y j n should it be easy or difficult, should not be very difficult look at the diagram out here ((Refer Time: 35:03)). You see y k ultimately what we are doing is that v k we are getting, now v k will be nothing but, expressible as the summation of y j w k j. So, this should be computable, so we should be able to compute this, we should be able to compute dou e k dou v k very easily.

And then we will be in opposition to calculate dou E dou y j and if we calculate dou E dou y j we will go back to equation number 7 and we will be calculating delta j. And what is delta j delta j is nothing but, the local gradient of the hidden layer neuron j. And if we can calculate delta j for the hidden layer, then simply ((Refer Time: 35:58)) we will be applying this rule again, in order to adjust the weight between jth hidden layer neuron with it is preceding ith neuron in the previous layer.

So; that means, to say that we will be then in a position to adjust the weights prior to this also. Because, you see if we look at this diagram ((Refer Time: 36:24)) what we can do is that, we will be in a very easy situation, it will be quite convenient for us to calculate all this change of weights in the output layer. And then, we will go back and we will be calculating all these change of weights which are happening in the layer previously.

So, just simply what we have to do is to extend this mathematical analysis. So, we have to now concentrate on this dou e k dou v k computation and dou v k dou y k computation. We can do that let us call this has equation number 9, that is dou E n dou y j n equal to this, so if we can compute all the components of equation number 9. Then, we will be in an easy situation to compute the local gradient of the hidden layer neuron.

(Refer Slide Time: 37:30)



© CET
I.I.T. KGP

$$e_k(n) = d_k(n) - y_k(n)$$
$$= d_k(n) - \phi(v_k(n)) \quad \ldots\ldots\ldots (10)$$
$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\phi'(v_k(n)) \quad \ldots\ldots\ldots (11)$$

For neuron k,

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n)\, y_j(n) \quad \ldots\ldots (12)$$
$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \quad \ldots\ldots\ldots\ldots (13)$$

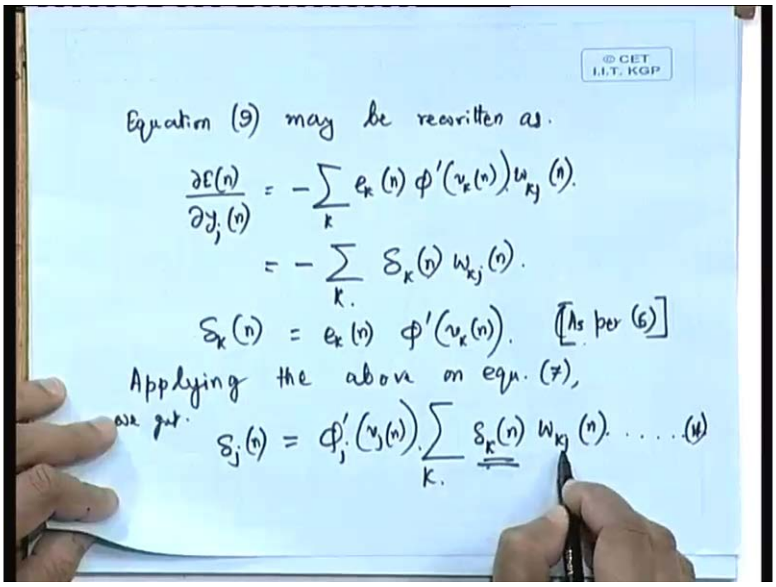
So, let us see that dou E k dou v k it is going to very easy why because e k n is simply equal to what d k n minus y k n and y k n is nothing but, phi of v k n. So, what is dou e k n dou v k n that is equal to minus phi dash, in fact we should be write phi dash of k anyway phi dash of v k n. So, this one we call as let us call this one as equation number 10 and this one as equation number 11.

Now, for neuron k the induced local field can be written as v k n equal to summation j equal to 0 to m w k j n y j n what is m in this case number of hidden layer neurons means ((Refer Time: 38:46)), v k I assume is connected to m number of neurons to it is preceding layer. So; that means, to say that they are m number of neurons in the hidden layer just preceding to the output layer, so this is expressed as j is equal to 0 2 m, 0 there are n number inner layer neurons.

So, leave it j is equal to 0 cases for the bias, so this is equal to v k n is equal to summation of this, which should be our equation number 12. So, from this it is easy to calculate dou v k dou y j n and what is that, that is equal to w k j of n very good. So, this is our equation number 13, so have we go what we were want it I think, so we have got dou e k dou v k we have got dou v k dou y k. That is, what we were wanting is not it dou e k dou v k and dou v k dou y j these two terms we wanted.

So, now that we know that very well dou e k dou v k being equal to minus phi dash of E k j and dou v k dou y j being equal to w k just simply, we can express now dou E n dou y j n. That is equation number nine can now be rewritten as...

(Refer Slide Time: 40:24)



© CET I.I.T. KGP

Equation (9) may be rewritten as.

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = -\sum_k e_k(n)\,\phi'(v_k(n))\,w_{kj}(n).$$

$$= -\sum_k \delta_k(n)\,w_{kj}(n).$$

$$\delta_k(n) = e_k(n)\,\phi'(v_k(n)). \quad [\text{As per } (6)]$$

Applying the above on equ. (7),

we get.

$$\delta_j(n) = \phi_j'(v_j(n)) \sum_k \underline{\underline{\delta_k(n)}}\,w_{kj}(n). \;\ldots\ldots (11)$$

So, I can do this equation 9 may be rewritten as dou n dou y j n which is equal to minus of the summation of e k n phi dash v k n w k j of n all right and just tell me that what is this minus of e k phi dash v k n delta k. So, this is equal to now this e k phi dash term is equal to delta k, so this is equal to minus of delta k n times w k j n and I have to sum it of over k.

So, this is interesting, so we have got the dou E dou y j and in this case we have made use of the fact that delta k n is equal to e k n into phi dash v k n going by our original definition. In fact, we had a possibly number that also yes. That is, as far equation number 6 we have express this as follows, so now we go back to the expression involving delta j all right we now want to compute delta j. So, delta j is equation number 7.

So, we have to go back to equation number 7, so applying the above or equation 7, we get what, we get delta j in fact, we should write delta j n itself. That is equal to phi dash of j into v j n the term ((Refer Time: 43:13)), that is already existing with us phi dash j v j n and we have made dou E n dou y j n. So, dou E n dou y j n is simply this, so simply this expression.

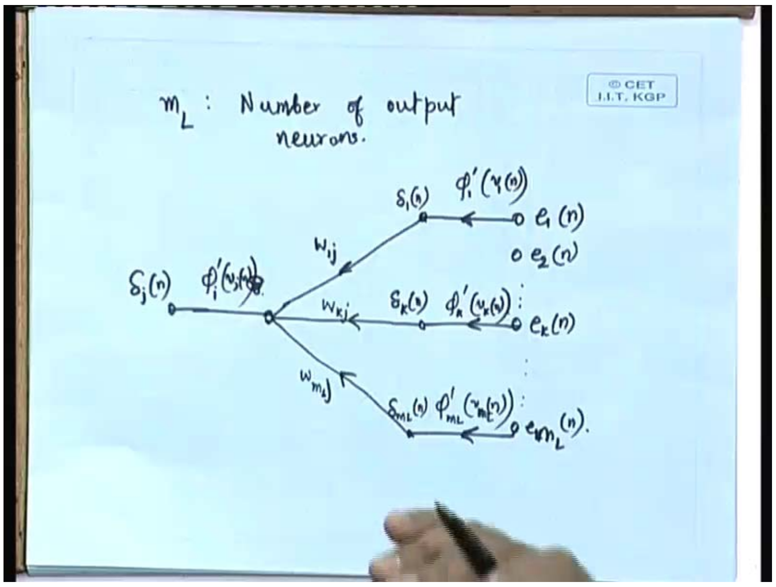
So, since we already had a minus over here and this one is also giving us this dou E n dou y j n is giving us another minus sign. So, together it will be of positive sign summation of delta k n w k j n and this we have to sum up over k. So, we have got one very interesting expression, whereby we are expressing the local gradient of the hidden layer neuron in terms of the local gradients of the output layer neurons.

In fact, it is nothing but, the weighted sum of also all the local gradients of the output neurons to which it is connected. Because, here summation over k means, what summation over k means that this j is connected to all the case in the output layer. So, here what we are meaning is that, all the case since we are summing it up over all the case. That means, to say that whatever local gradients we have the output have at the outputs.

We are considering the summation of that weight summations over all that. And then, we are multiplying it by the derivative of the activation function of itself, that is phi dash j means, it is the derivative of the activation function of jth neuron. And then only we are getting delta j, so to compute delta j this signal flow that we are getting is very interesting let us see.

You see, that what we are doing, we are calculating all these deltas, deltas are nothing but, E times phi dash is not it deltas are nothing but, E times phi dash according to our definition E times phi dash. So, what we can simply show is that, let us say that we have got number of output neurons are there.

(Refer Slide Time: 45:52)



Let us say, that we have m L as the number of output neurons. So, what we are showing is that here, we are showing all the errors, let us say that this is e 1, this is error e 2, this is error e k everything at the nth iteration. Finally this is our e m L n, so for every output neuron we can calculate the error terms is not it. And then, if we multiply the errors by their respective activation function derivatives.

In that case what we are trying to do, this E k is this E's are available to us and we are multiplying that by E 1 is available in this case. So, E 1 will be multiplied by phi dash 1 v 1 n and what will I get as this term delta 1 n, like that e 2, like that e k n. So, here for here when e k n is the error term and we are multiplying it by phi k dash v k n. Here, what are we getting as delta k n and the last term will be here phi dash m L v k n or here v m L n.

And here we are getting delta m L n, so all these deltas are available now. Now, what we are doing, again go back to this equation ((Refer Time: 47:52)) study this equation what should we number this, we should give the number as a, but what is running number, what was the last number that we have given 13.

Student: 14.

14 equation 14 we use. So, then what can do is that all these things we have to add up, so multiplied by the.

Student: ((Refer Time: 48:16))

Now ((Refer Time: 48:20)) this delta k’s we have got they have to be multiplied by w k j’s. So, what we are doing that a first term will be multiplied by w 1 j, the kth will be multiplied by w k j and the m Lth term will be multiplied w m L j all right. And this together will give me what, this together will give me delta j of.

Student: ((Refer Time: 49:00))

No, they are not they are not. So, this is simply summing it up and then this summed term we have to go through what, phi dash j v j of n and that will give me delta j n. And here we should indicate the signal directions also, if we are first starting with the error terms multiplying the errors with the derivatives, with the derivatives are the activation functions getting all the local gradients multiplying the local gradients by the synoptic weights, synoptic weights from the jth neuron to all the output neurons.

So, synoptic weights we are doing the multiplication, getting the sum of all these weighted sums, that is to say sum of delta k w k j’s. And then, on this summation we operate the phi dash term and then we are getting the delta j, which is the local gradient of the jth neuron. And once we get the local gradient of the jth neuron the calculation of the weight correction is pretty simple. In that case, what we simply have to do is like the way we had applied, you remember simply this equation we should reapply ((Refer Time: 50:57)).

So, where delta w j i will be the change of the weight, that we are doing between the jth neuron and the ith, ith is connected to the layer previous to the hidden. So, delta w j i n will be equal eta times this delta j n, but this delta j n is going to be the local gradient of the jth neuron, times y i n and yin is the input is the ith input to the hidden layer neuron j.

So, this basically completes our analysis on the that propagation algorithm. In fact, this is in an nutshell ((Refer Time: 51:44)) what the back propagation algorithm looks like. You see all the mathematics that we have done. In fact, they are all very simple mathematics, because the only great thing that we have done is taking the derivative and that to instead of taking complicated derivatives.

What we have done is simply to conveniently apply our chain rules. So, that the derivative expressions become as simple as possible and after doing all that, what results is that, you start with the errors, multiply the errors with the activations get the gradients. Multiply all the individual gradients with the synoptic weights, get the gradient of the preceding layer. And this time j is just directly connected to the output, but what happens it j is sandwiched between another hidden layer.

Supposing in general j is sandwiched between two hidden layers, then also there is no problem. Because, once we know this delta j n it is possible for us to propagate this delta j n backwards also. So, the error term begins with the output and the error term propagates from layer to layer backwards. And what is the last layer up to which we have to go, where have to go up to the layer just after the input.

So, in effect what we are doing in a back propagation algorithm is that, there is a forward pass and there is a backward pass. Forward pass does what, it will take the inputs first compute the induced fields and the outputs at the layer just next to the input, that should set a first hidden layer. Then, from first hidden layer it goes to the second layer, then from the second layer it ultimately goes to the output layer.

Output layer all the outputs are available, it calculates the error terms and they are forward pass ends, you forget about the forward pass. Now, the backward pass begins we start with the error terms, we calculate the local gradients, just propagate those local gradients in terms of the errors only, just propagate those local gradients into the layer next to the output, layer previous to the output.

Propagate it once step one layer before that and like that you keep on going ultimately you reach the layer just next to the input. And then, everything is adjusted, all the synoptic weights, not only the once that is connected to the output. But, all the synoptic weights which are connected to the previous layers, they are all adjusted once for all.

Only underlying assumption in this case is, that during this time that is to say when the forward pass and the backward pass goes on during that time our input should be held straight. Because, this computation may take some time and if during that time that input changes, then we will be having problem. So, the underlying assumption is that during the computation of the forward pass and the backward pass, the input the first level input must remain straight line all right.

So, that is all for today’s lecture and tomorrow we will bring in some more aspects pertaining to the back propagation algorithm.

Thank you very much.