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

Lecture - 34
Hebbian - Based Principal Component Analysis

Today's topic is Hebbian Based Principal Component Analysis.

(Refer Slide Time 00:55)

Now, this primarily follows from one of the very important results that we had presented without giving the proof. That is to say that in order to realize a principal component analysis computationally. Rather than going in for the classical mathematical way of solving the Eigen value problem. And then, finding out the corresponding eigenvectors to it and projecting the input vectors into those Eigen space.

The same computation can be achieved by using single neuron which does, which is based on a Hebbian mechanism of learning. So, Hebbian mechanism of learning as you very correctly can recall, that it says that if the pre-synaptic signal and the postsynaptic signal they are correlated. If they are both of the same sign, in that case the synaptic weight has to be updated, increased in that direction.

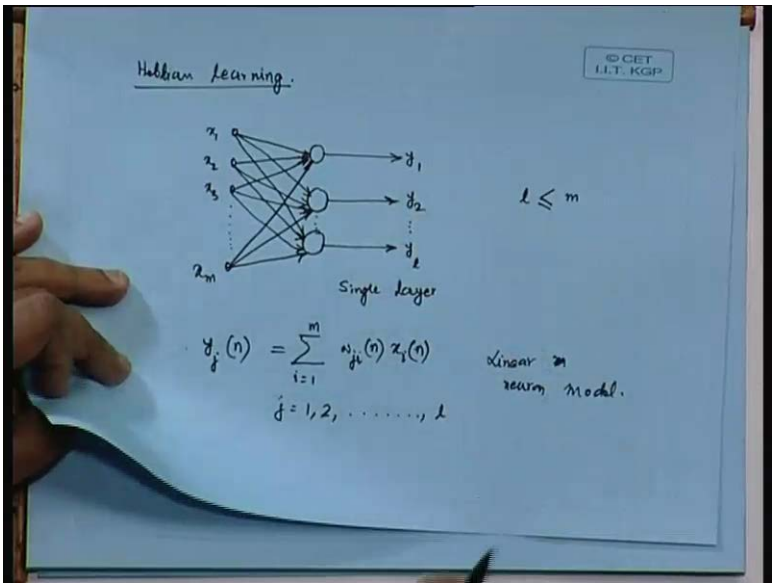
So, that next time when the same condition occurs, then with a greater strength it can achieve the same result in the pre-synaptic and postsynaptic conditions. So, a single neuron can basically do the, basically compute the principal component analysis of the first component. First component means that, if we are arranging the Eigen values in the decreasing order, then the highest Eigen value that results. We call it is as lambda 1 and corresponding Eigen vector we call it as the q 1 vector.

And it is the projection of the x vector into the q 1 vector, which we are referring to as the first principal component. Now, first principal component as we said can be computed using linear neuron that, works on the Hebbian learning mechanism. And here what we are trying to do is that, just to extend that concept into the first one principal components because, essentially that is what can lead to some practical applications for us. Because, nobody will be interested in knowing just the first principal component alone, all though that, is the most dominant one.

But, in a very typical problem where there is a data reduction, now certainly one would like to keep some amount of data at the output also. So, if we have got m dimensional input, we are looking for an l dimensional output, but l is less than or equal to m, but certainly we do not want l to be equal to 1 exactly, because 1 is too much of a reduction. So, we want to keep some that, means to say that whatever solution one can achieve using a single neuron, one has to extend that.

So, that the first l principal components could be extracted, so to do that we have to use a network of this nature where at the input we will be having m number of input basically at the input layer.

(Refer Slide Time: 04:52)



So, we mark these inputs as x 1, x 2, x 3 up to x m, so that together they will form an m dimensional x vector. And let us say that we have got one different number of computational units or neurons, so we are taking one of them. So, that we will be generating one such outputs from those neurons.

So, we will be having y 1, y 2 etcetera up to y l as the output and our l is going to be less than or equal to m. Typically less than m only that is what we will be having, if we are going in for a data reduction. So, now what we have to do is to make the connection from this input layer to the output layer and it should be typically fully interconnected type of a network, between the layers.

So, that what we will be doing is just to connect all these x inputs to each of these neurons. So, we will be having m connections like this, again very similarly we will be having m connections for the second one, like this we continue and m connections for the last one. And since all the neurons they are linear, so this is basically a single layer, so you can see that we are using a single layer of neurons.

And the output of those neurons could be written like this that, if we are considering the j th neuron output at the iteration number n. So, we are representing that as y j of n indicating that the suffix j is the j th neuron. And this n the number written in the parenthesis is the iteration number, the iteration of learning. So, this y j n would be in the

expanded form written as summation of i is equal to 1 to m this corresponds to all the inputs.

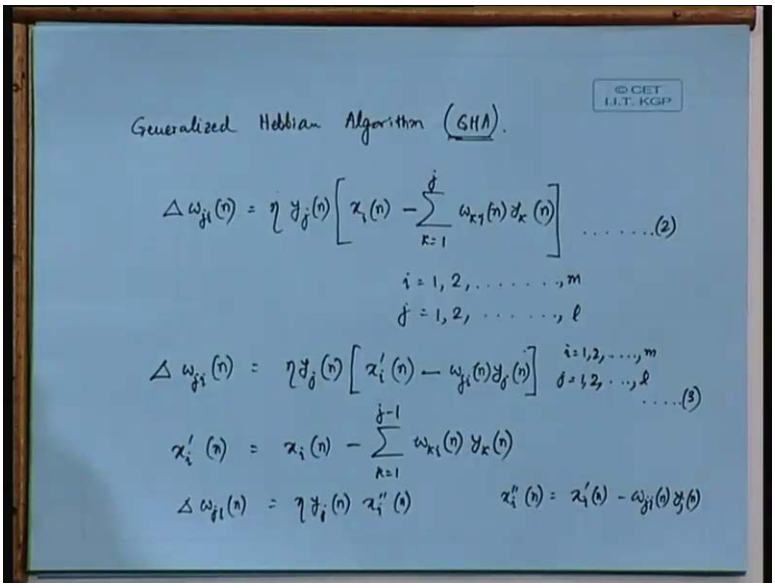
And it will be the summation of the w j i at the nth iteration, so basically the weights, the synaptic weights are actually arranged in the same way. That means, to say that when we are taking the j th neuron over here, then by w j i. We will be referring to the synaptic weight connection from this j th neuron at the output. In between the j th neuron of the output and the i th input to it, so it is w j i x i of n. So, it is a linear neuron model, so what we have shown is a linear neuron model.

Now, in this case what happens is that the synaptic weight w j i n, that will be updated in accordance with the Hebbian learning mechanism. Because, we are all the time assuming that the learning mechanism that we are following is Hebbian learning. Now, in this case what one has to consider is that, if you are considering the synaptic weight updating of w j i, that means to say that whatever is connected to the j th neuron.

If you are considering, then all the synaptic weights that are connected from the i th input to the neurons, which are previous to this j, the earlier to those j that is also to be taken into consideration. Because, what happens is that, Hebbian learning is a mechanism of a positive feedback type, that means to say that you can imagine it like this that. Since it is a combination of the pre-synaptic and postsynaptic weight. So, x 1 and y 1 together activating actually strengthens w 1 1.

And again since that strengthens y 1, so that is why when you are considering x 2 with y 1, the connection of that gets the w 2 1 it gets altered. And accordingly what happens is that one can formulate a generalized Hebbian algorithm. Now, we are not going into the detailed analysis of the generalized Hebbian algorithm. But, I am just presenting you the result of the generalized form of the Hebbian learning.

(Refer Slide Time 10:40)



So, what we are writing now is a generalized Hebbian algorithm and in the short form it is written as GHA. And according to GHA the change of the synaptic weights, because here we are taking the j th neuron output. So, we will be interested in finding out that what is the delta w j i, the change of synaptic weight. So, according to the generalized Hebbian algorithm, the change of weight that is delta w j i for the iteration n would be written as eta, eta being the learning rate multiplied by...

We did not write down one thing, for the sake of completion we should have written down here ((Refer Time: 11:36)), that this neuron model should be followed for all the j's. That is to say for j is equal to 1, 2 up to l, because here there are l. And in this case the equation that we are going to write will be equal to eta times y j n, had it been a single neuron we know that we would have considered eta y j n x i n. But, according to the generalized form of Hebbian learning, this is not exactly x i n, we have to write it as eta y j n x i n.

But, from this we have to subtract a term which is given by the summation for k is equal to 1 to j w k i for iteration n y k of n, so have you followed what we have considered here. You see if you try to interpret it you are considering the j th neuron, now j th neuron and j th output and i th input. So, for the i th input yes this x i of n is the term that is contributing, but apart from this term there are some other terms which will be having it is effect and what are those terms basically they are the summation of w k i y k.

Now, this w k i y k will mean that, we are considering the k th output and summing it for k is equal to 1 to j and k is equal to 1 to j means that, whatever has already occurred for this. So, all the previous y k's are been taken into consideration, because they will be accordingly modifying the effective x i n. So, this thing can be regarded as the effective x i n, because of all the contribution of these terms. And here in this equation our i will vary from 1 to m and our j the output index will vary from 1 to l.

So, this we are calling as equation number 2 and the first one that we had got we are calling that as the equation number 1. So, this is the change of the synaptic wave that we are going to have and we are just going to write it in a better form in the sense that, instead of writing it like this. Let us write this equation number 2 in this way, w j i n will be equal to eta y j n. And let us define a quantity x i prime n, where basically this x i prime n we are defining as x i n minus summation k is equal to 1 to j minus 1 means all the previous.

In fact, this include j mind you here this generalized Hebbian algorithm, that includes this j th term. Whereas what we are considering as the definition of this x i prime n is that it is x i n minus k is equal to 1 to j minus 1, so we are going one before, w k i n y k of n. So, if we define the x i prime like this, then we will be left with some other term over here, so what will be after this, what will be x i prime n.

Studnet6: ((Refer Time: 16:00))

Minus obviously, because all this terms are minus, minus w j i n y j n, so what we are essentially doing is that we are separating out the j related term. The j th neurons term we are just separating out of this summation and all the other things we are embedding into the definition of this x i prime n. And in fact, this itself is acting as a modified input or so to say x i prime n minus w j i n y j n this entire term that we have written within this square bracket, that is basically serving as the modified input term.

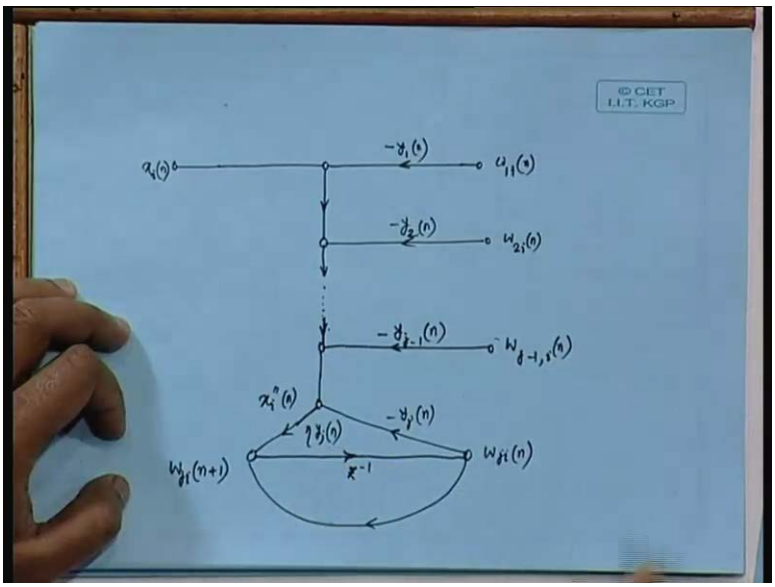
So, we can rewrite this equation call this thing, so this also is for i is equal to 1 to m and j is equal to 1 to l, so this we call as equation number 3. So, equation number 3 can be rewritten as delta w j i n equals eta y j n x i double prime n. So, now, you get your very familiar Hebbian learning term and here what is this, this x i double prime n is equal to x i prime n minus w j i n y j n.

So, basically what we have done is that, the term that is written within the square bracket, it is being substituted by x i double prime n which we have just now defined. So, this being the modified input the delta w j i is equal to eta y j x i double prime n. So, this will be the situation if we are looking for a generalized Hebbian learning, where we will be having m inputs and l outputs.

Now, this computation in fact, can be depicted in the form of a signal flow like this. You see that if we want to get the first term in this expression we have to of course, start with x i of n. So, we have to start with x i of n considering the i th input and from x i of n, we have to basically subtract all this j minus 1 terms that is coming out. So, what we have to do that, at first we have to subtract for k is equal to 1 term.

So, what we have to do is that from x i, we have to first subtract w 1 i n and y 1 n, w 1 i n multiplied by y 1 n and this we have to subtract from x i o f n to start with and then, we get a result. The difference of that, from that we will be substituting the term w 2 i n y 2 n, from that we be subtracting w 3 i n y 3 n, so it will follow in a flow like this. So, this equation let us call it as equation number 4. We can represent this equation 4 in the form of a signal flow graph and the signal flow graph for us would look like this.

(Refer Slide Time: 20:09)



That we will be taking x i of n and then, from here we will be first subtracting the term w 1 i of n and this is to be multiplied by minus y 1 n. So, this will lead to the first subtraction, so we get the first subtraction result the first term subtraction corresponding

to k is equal to 1. And what we get here, from that we have to subtract w 2 i n and this is to be multiplied by minus y 2 n.

For the other inputs it continues like here it will be w j minus 1 comma i n, so all the j minus 1 earlier neurons responses we are considering together, so here this will be minus y j minus 1 n. So, again you can verify that, all that we have done is the computation of this term with respect to equation number 4. So, with respect to equation number 4, all this j minus 1 terms are getting computed and in effect here what we are having, because already x i n has been fed over here.

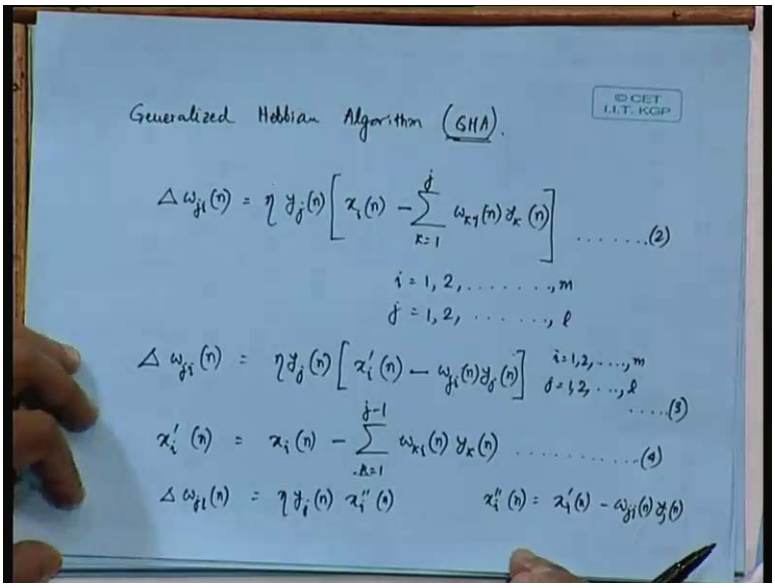
So, we are getting the x i prime n computed out of this signal flow diagram that we have drawn. And from here what we have to realize is x i double prime n, because x i double prime n we have said, that it should be equal to the x i n minus w j i n y j n. That means, to say that from here we have to subtract another term and that is w j i n y j n and it is being subtracted here.

So, what we have is that, we are just drawing the signal slope from here, so here mind you we have to give an arrow. And here we will be having w j i n and this to be multiplied by minus y j n and then, if we are subtracting this from this term, that is x i prime n we will be getting here what x i double prime n x i double prime n. And then, we can see that x i double prime n, if we are multiplying by eta y j that should lead to delta w j i n and what is delta w j i n delta w j i n is nothing but, w j n plus 1 minus w j i n.

Now, we already have w j i n over here, so what we do is that from this we take this as the connection. So, this is w j i n plus 1, in fact, w j i n plus 1 and w j i n will be related by this by 1 unit delay, because if you introduced 1 unit delay between this and this. Then what results is basically from w j i n plus 1, we can get w j i n by introducing the delay, so that is the relation between n plus 1 and n.

But, what we are doing here is that we are subtracting w j i n, from w j i n plus 1. So, in effect we are getting delta w j i and this delta w j i is what we are getting from this x i double prime n multiplied by eta y j n, so this is the signal flow graph of this equation.

(Refer Slide Time: 24:49)



The signal flow graph of equation 4 is obtained like this, now once again I repeat here in case you have any difficulty in appreciating what is generalized Hebbian algorithm has to tell as. Now, we have this delta w j i n eta y j x i n, this equation is very familiar to us we had already seen that in this case of Hebbian learning. But, mind you there we were considering only one output, whereas this one is generalized into, here the Hebbian learning mechanism has been generalized into more than one outputs.

So, this form we just accepting, since we are not going into the detail mathematical analysis of how it is being derived. I only try to explain the things in a more subjective fashion without going into the analysis part of it. So, this flow basically realizes the equation that we were describing. But, what is going to be interesting to us is that to show that this network is actually extracting the principal components one after the other.

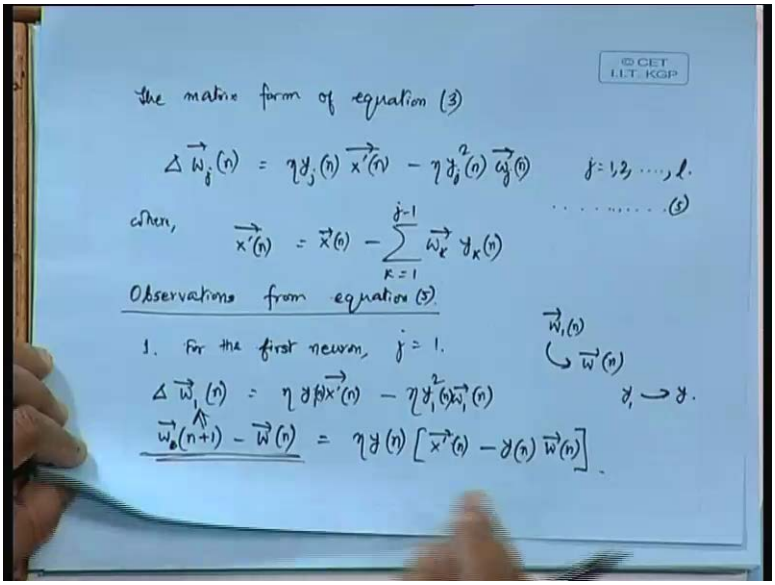
That means, to say that if we take the first neuron that will extract the first principal component. The second will have the second principal component and like that, so we can show that in a very simplified manner. So, before we can show it, let us try to rewrite this equation number 3, this equation number 3 will be defined for i is equal to 1 to m and for j is equal to 1 to l.

So, if we repeat this equation number 3 for all this i's, that means to say that if we write down m such equations, then what results is that we will be having w j 1 n equal to this

w j 2 n, like that we write m different equations. And writing those m different equations would be equivalent to expressing that in a matrix form. So, what we do is the in the form of matrix we will now represent this equation number 3 taking the expansion j in to consideration.

That means, to say that we will now be defining w j 1, w j 2, I mean delta w j 1, delta w j 2 up to delta w j m and together we will be calling that as delta w j vector.

(Refer Slide Time 27:51)



So, the matrix form of equation 3, the matrix that can be written as delta w j n and this is equal to eta y j n an x prime n. If we write it in this form, not equation 3 let us write in the form, yes in equation 3 only we have got x prime term. So, it will be x prime n again defined in the form of vector because this is x i. So, we have to consider i is equal to 1 to m, so this becomes x i vector minus this becomes you can say eta y j square and w j n will again become a vector.

So, this is equal to minus eta y j square for iteration n w j n and this is for j is equal to 1 2 up to l. Where in this equation x prime n as a vector is equal to the definition of equation number 4 we can simply apply. Again this equation number 4 also is to be written for all the i's, so this is valid for i is equal to 1 2 up to m. So, we can write this equation number 4 in the matrix form as x i prime n vector equal to x vector minus summation k is equal to 1 to j minus 1 w k vector y k n. So, this is the matrix form of this equation.

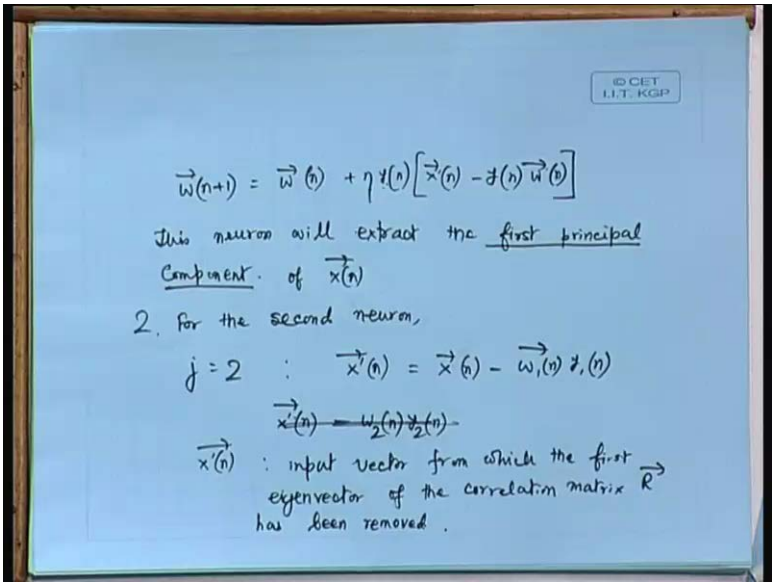
And now we can make the following observations from this equation, if we call this as equation number can we call it as 5.

Student: ((Refer Time: 30:12))

Yes correct, thank you for pointing it out. And now we can make some observations out of equation 5. Let us take the first neuron, so for the first neuron, the index j that we have to consider is j is equal to 1. So, we try to write the j equal to 1 case, now j is equal to 1 case will be of course, delta w 1. So, delta w 1 of n can be written as w 1, we are not writing 1, because we are considering only this j is equal to 1 case, we could write if you want we can write w 1 delta w 1 n.

Now, delta w 1 n will be equal to let us write down this thing first it is equal to eta y 1 x prime n and then, we have got what minus eta y 1 square w 1 n. Now, instead of w 1 n we can write as w n, so we interchangeably write w n for w 1 n. And then, since it is delta w 1 n this is one and the same as saying that it is w 1 n plus 1 or w n plus 1 minus w n. So, if we write down this for w 1 n, then what results is as follows.

(Refer Slide Time 32:28)



We can rewrite this as w n plus 1 is equal to w n plus, that means to say that this minus this is delta w, delta w n. So, what we are doing is that, this is equal to this eta term, so after eta we were having lot of terms eta, now we are going to call this as eta y. So,

instead of w 1 n, we are calling it as w n and instead of y 1 we are calling as y. So, this whole thing can be written as, here also we should have taken eta y n.

So, this can be written as eta y of n and then, within the bracket we will be having x prime of n minus y n w n, so this is this. So, that our w n plus 1 simply bringing this w n term from the left hand side to the right hand side. What results is w n plus eta y n and within the square bracket we will be having x prime n minus y n w n, so this is the first neuron. And in fact, this can be proved in fact, I was yesterday referring to OZAS result, OZAS of the paper which he published in 1982.

Now, according to his result he has proved that, this neuron will extract the first principal component. So, if that is the case, then assuming that the first neuron has already extracted the first principal component. Then for the second neuron we can see that, what happens what is the thing that the second neuron extracts. So, for the second neuron obviously, there we have to take j is equal to 2 and if we take j is equal to 2, then we can see that we have to take this as x prime n, for the second neuron will be equal to x of n.

And from there we have to subtract w 1 n y 1 n, so the term that we have already considered. So, there basically what we have to do is this minus this has been already accounted for the first neuron, so that will be our x prime n for the second case are you getting it. You consider j is equal to 2 again look at the equation number 5, you consider j is equal to 2, j is equal to 2 means that you are getting w 2 n plus 1 minus w 2 n and that becomes equal to eta y 2 n.

And then, within the bracket we will be having the term as x prime n minus we will be having this for y 1 n w 1 n term will remain and then, there will be minus y 2 n w n. So, we are considering that the new expression for x prime n, for j is equal to 2 is this term, from which we have to subtract y 2 n w n and y 2 n w n is the thing which we are now going to refer as the new y n w n. Anybody having any difficulty in following the result, see from here we just have to subtract this term.

So, if we consider x prime n minus w 2 n y 2 n, then that is the same thing as this term. So, here the new x prime n is become in x of n minus w 1 n y 1 n. So, what is this, this is the case that from the input vector, this x prime n is basically realizing an input vector from which the first eigenvector of the correlation matrix has been removed because, we

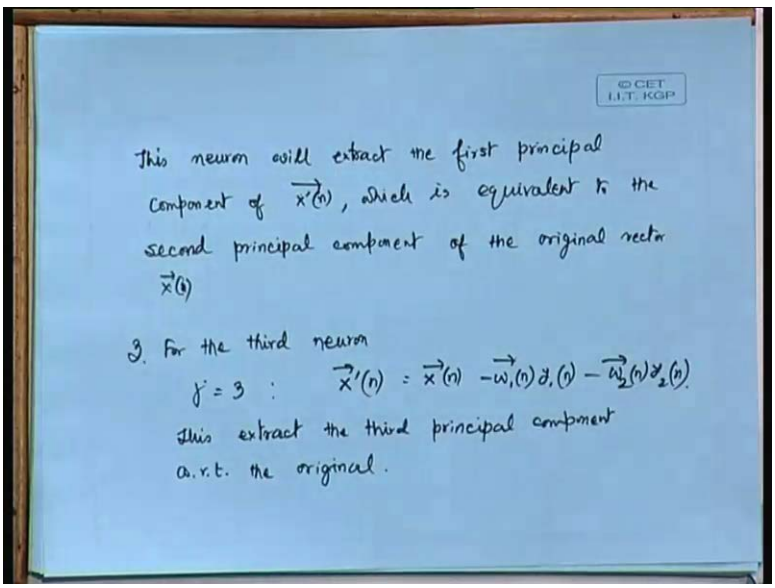
are taking this result that this neuron has already extracted the principal component, first principal component.

So that means, to say that x n x prime n, now becomes a modified input vector from which the first principal component has been eliminated. Or rather to say the first eigenvector of the correlation matrix has been removed. So, x prime n, our new x prime n for the second neuron is basically the input vector minus the first principal component already taken out.

So, now this modified input vector will be applied on this algorithm again. So, what we will go through now is simply w 2 n plus 1 will be equal to w 2 n plus eta y 2 n x prime n x prime n is this 1 minus y 2 n w 2 n instead of 2 you can write it as y and w. Instead of y 2 and w 2, you can write as y n w, because we are interested now in extracting j is equal to 2. And what will it extract it will extract the first principal component of x prime n, earlier it was extracting the first principal component of x n.

And now it is going extract the first principal component of x prime n. And first principal component of x prime n means, that it becomes a second principal component with respect to the original, with respect to the original vector.

(Refer Slide Time 40:43)



So, here this neuron will extract the first principal component of x prime n which is equivalent to the second principal component of the original vector x n. And now we can

continue this iteration, so if we now take the third neuron. So, for the third neuron definitely we have j is equal to 3 and there what will be x prime n, what will be the x prime n for the third one, anybody can suggest.

Student: ((Refer Time: 42:21))

The first two terms will go away from the x n vector, so x prime n will be yes, equal to x n.

Student: ((Refer Time: 42:33))

Minus w 1 n y 1 n minus w 2 n y 2 n, so from the third neuron. So, that means to say that his will extract the first principal component of x prime n equivalent to second principal component with respect to the second neuron or rather to say the third principal component with respect to the original. So, this extracts the third principal component with respect to original, so this we will continue up to all the l.

This is one way of showing that we are considering one after the other. But, that does not make this algorithm sequential mind you, it is only for the ease of the analysis that we have shown it like this. That considering the first one to have already extract in the first principal component, how the other neurons extract at the same way the second one, third one like that, we had shown in a step, but everything can be trained together.

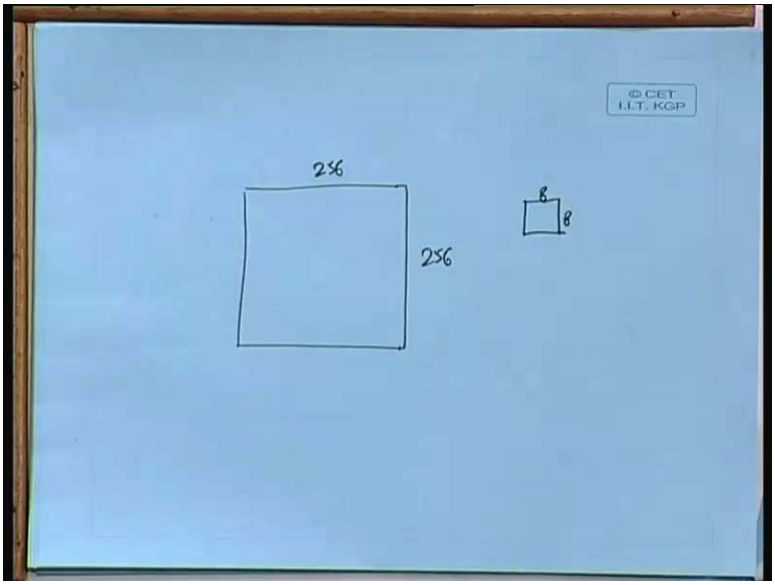
So, when the network having l neurons will be trained by a Hebbian mechanism with sufficiently large number of n. In that case, that will ultimately evolve into the extraction of the first l principal components. And this kind of a mechanism will you call it as a supervised or a unsupervised.

Student: ((Refer Time: 44:47))

Definitely unsupervised, because we are not feeding any target output for this. We do not have any desired or target output for that, and basically what we are doing is that, we have to keep on feeding the different input patterns. We have to feed various examples of this x n vector and if you feed all this different examples of x n vector. Then the network will train up finally, train up in the sense that it will be finally able to extract the first l principal components. And this has got a very wide application.

So, I can make a mention about the image coding application, where the principal component analysis can be employed. So, primarily what you need to do is that, supposing you are given with an image.

(Refer Slide Time 45:52)



Let us say some image of a fixed size, let us say that you have got 256 by 256 pixels of the image. Now, what you can do is that this image you have to represent in fewer number of bits, that means to say that you have to achieve some data compression out of it. Because, if you look at the original image, then it will be containing 256 by 256 number of pixels. And even if you consider it to be a gray scale image, then it will be 256 by 256 by 8 those many number of bits will be required to represent that.

Now, what you can do in order to achieve data compression is that, you can divide this 256 by 256 into some non overlapping blocks into non overlapping blocks of 8 by 8 or some size like that. I mean let us say that we divided it, subdivide the image into non overlapping blocks of size 8 by 8. Now, 8 by 8 basically corresponds to 64 inputs, 64 pixels.

So, considering those 64 pixels as the input, we can train up a principal component analysis network. Or rather to say a neural network like this, which will be having 64 inputs and less than that number our desired number, if we want those many number of outputs we will be having.

So, if we train it up with sufficient examples of this 8 by 8. In that case the network will ultimately extract the first l principal components of that. And the first l principal components that we are getting will represent the original 8 by 8 block. So, like that every block can be transformed into a block to a data space which has got a dimension less than that of 64, in this case it is 64.

So, we are going to project it into a space and we are going to get it less than that of 64. And then, if we apply the reverse transformation, because we know that how the synthesis also can be carried out. The synthesis also can be done using the neural network structure with Hebbian learning and then, we can obtain the reconstructed image back. It is the same encoder and the decoder mechanism that I was telling you in the last class, that where the encoder will represent it in a fewer number of samples as compare to the original input space.

Whereas, the decoder will restore it back, only thing is that the decoder will not be able to restore it exactly, because we have already decided to chop of the data in the projected space. And projected space truncation of the data will not contribute to much of an error, because we have projected it into a space so judiciously, that only the low variance terms are eliminated.

So, that is essentially the purpose and the use of the principal component analysis. And neural network with Hebbian learning mechanism contributes to a better practical realization of the principal component analysis. You have got any questions, if there is none, then we can stop it for the day. And then, we will be taking up the next topic as a comparative learning mechanism which is the self organizing map. So, in the next few lectures we will be talking about the self organizing maps.

Thank you very much.