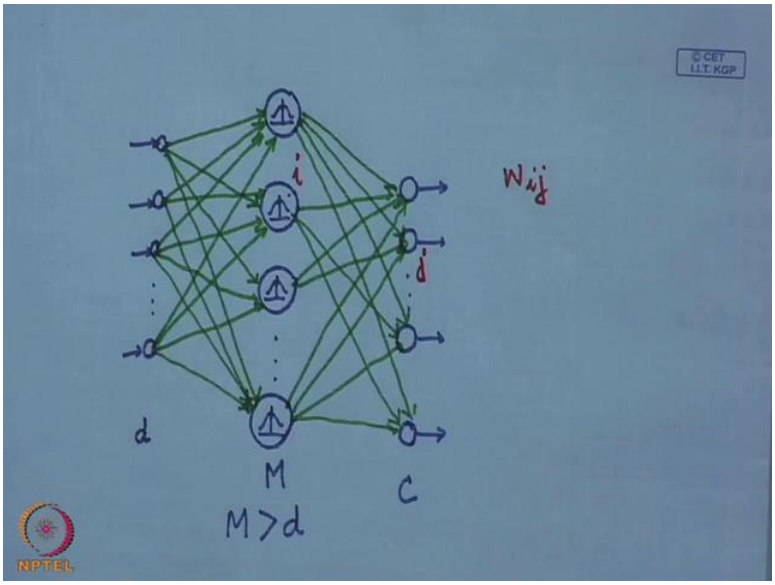
Pattern Recognition and Applications
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture No. – 28

Hello. So, in the last class we have started discussion on radial basis function neural network. We have seen that radial basis function neural network consists of three layers, 1 is of course, the input layer and 1 of the three layers is output layer and in between the input layer and output layer we have a hidden layer. So, unlike in case of multilayer perception, where we can have 1 or more hidden layers. In case of the radial basis function network we have only 1 hidden layer, and every neuron in the hidden layer computes a radial basis function.

So, when I have the neurons in the hidden layer. For every neuron, which computes a radial basis function, radial basis functional value for an input feature vector. Every bit radial basis function has got 2 parameters, 1 is called the receptor and other one, which defines the trade of the radial basis function. So, the architecture that we have is something like this.

(Refer Slide Time: 01:49)



We have 1 input layer, so the input layer contains a number of neurons and the number of neurons in the input layer is same as the dimensional, dimensionality of the feature

vector. So, that if the feature vectors of are of dimension d, I will have d number of nodes in the input layer so there will be d number of nodes. When the dimensionality of the feature vector is d in the hidden layer I will have a number of nodes.

And suppose the number of nodes in the hidden layer is M. So, as we discussed in our previous talks that the purpose of the hidden layer nodes is to project that d dimensional feature vector into a higher dimensional feature vector. So, as I have n number of nodes in the middle layer, so obviously this M the number of nodes in the hidden layer is greater than the dimensional. As we say it that every node in the hidden layer computes a basic radial basis function.

Like this and at the output layer, which are basically the classifying neurons, I have the number of neurons of the number of nodes, which is the same as the number of classes that we have. So, if I have c number of classes then at the output layer I will have c number of neurons.

So, there we have c number of neurons. Where c is the number of class in which the pattern has to be classified. Then every node in the input layer is connected is feeding input to every node in the hidden layer and output of the hidden layer every node output from the hidden layer is connected to every node in the output layer. So, I have the connections, which is something like this.

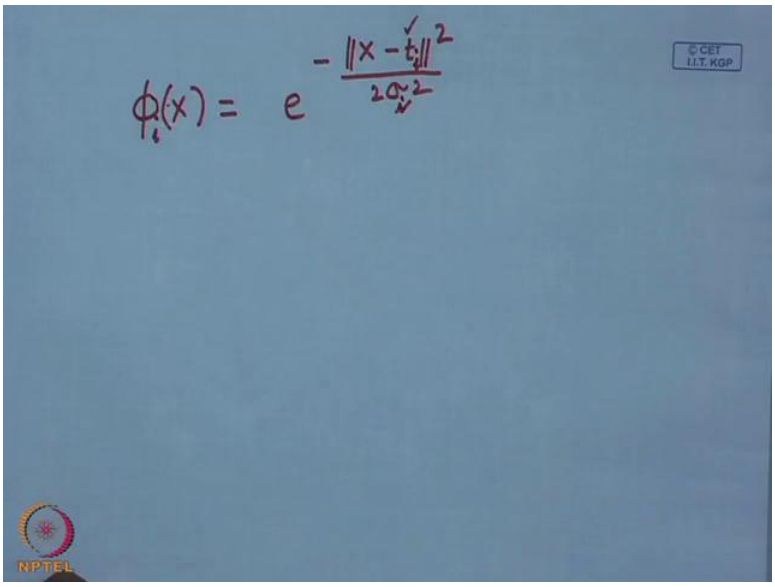
So, these are the connections from the input layer nodes to the hidden layer nodes. Because the purpose of this connection is simply to forward the input feature vector to the nodes in the hidden layer, we can assure the weight of each of this connection is equal to 1 and that is a difference with the connections from the hidden layer to the output layer nodes. Because, in every output layer node, computes a linear combination of the outputs of the hidden layer node. So, the connection from the output layer node to the connection from the hidden layer nodes to the output layer nodes is something like this.

Where we can say that every field i th node in the hidden layer is connected to the j th node in the output layer to a connection weight, which is equal to W i j. So, because of this every node in the output layer computes a linear combination of the outputs of the hidden layer. Based on this, the value of this linear combination the output layer nodes decides to which class the input vector should be classified.

Now, what can be d1 is these output and nodes can also impose a non-linear function to ensure that if a particular input feature vector belongs to class omega j. In that case only the output of the j th node will be equal to 1 and output of all other output will be equal to 0. Similarly, if a feature vector input feature vector belongs to say class 1 then only the output of the first node in the output layer will equal to 1 and outputs of all other nodes will be equal to 0.

So, as I discussed in the previous class that search a radial basis function network and r b f network incorporates 2 types of learning. 1 is, we have to learn that for every node in the hidden layer, because every node in the hidden layer represents a radial basis function, what should be the receptor of that radial basis function and what should be the state of that radial basis function.

(Refer Slide Time: 08:28)



So, if the radial basis function is a caution function, that is if it is something like this, say phi of a x is equal to say e to the power minus x minus t square of this upon 2 sigma square, where t is the receptor and sigma which is the variance. It decides that what is the spread of reduces. So, every for every i th radial basis function y i x t i is the receptor and sigma i is the straight, so I have to know that what is the receptor for every radial basis function and what is the spread of every radial basis function. So, this is 1 level of learning and the second level of learning is once through these radial basis functions our d dimensional feature vector is projected onto a balanceable feature vector.

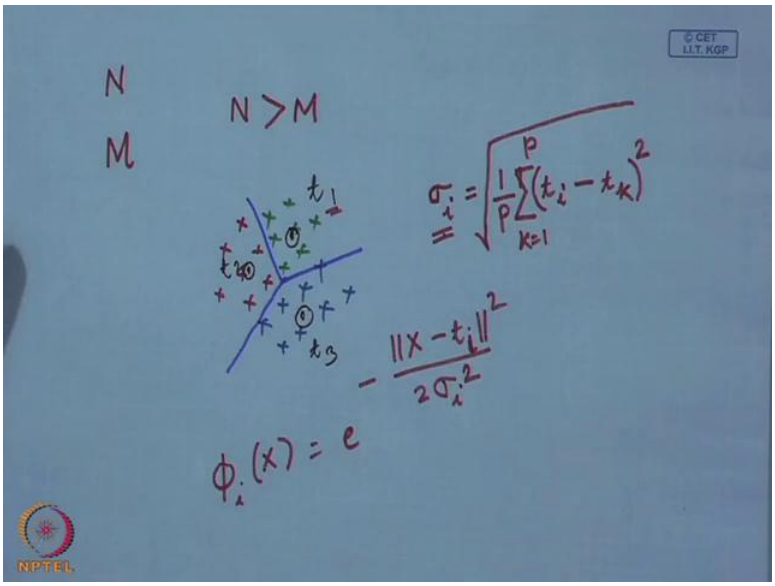
Basically, what we are doing is, we are increasing the dimensionality of the feature vector. As we have indicated in our last class that the purpose of increasing dimensionality is that if the feature vectors are linearly non separable in the determination of spires will cost them into higher dimensional space. Then the possibility that they will be linearly separable in a higher dimensional space increases and this possibility increases with the value of M. So, as we increase the dimensionality more and more, the possibility of linear separability of the feature vectors also increases.

So, the feature vectors in the d dimensional space, which am not linearly separable, when I asked them into an M dimensional space at M is greater than d, it is more likely that those feature vectors, will be linearly separable in an M dimensional space. Once the feature vectors are linearly separable in the M dimensional space, then the linear combination of the outputs of this ((Refer Time 10:53)) layers is likely to give me a cross belongingness. And that linear combination is decided by the weight vectors by the connection weights from the hidden layer nodes to the output layer nodes.

So, we also have to learn that what should be the connection weight W i j from the i th node in the hidden layer to the j th node in the output layer. So, this is the second level of learning, so in the first level of learning for every radial basis function, which are to learn what is the receptor and what is the spread of radial basis function.

In the second level, we try to learn what is the connection weight from the input layer from the hidden layer nodes to the output nodes. As we have discussed in the previous class that the usual way and common method of learning the radial basis function is if you are given a state of feature vectors of the training purpose. Suppose, value of M is equal to 3, so what we do is we partition weight of cluster the state of feature vectors into the number of clusters.

(Refer Slide Time: 12:29)



So, if we have M number of nodes in the hidden layer and I have say N number of feature vectors, N number of feature vectors, which are given for training purpose and I have M number of nodes in the hidden layer, obviously in this case N has to be greater than M. Otherwise, clustering N number of feature vectors into N number of clusters does not make any sense.

So, I have to have more number of feature vectors than the number of clusters that I have to form. So, I cluster this N number of feature vectors into M number of clusters and I can assume that centroid or mean of every cluster represent the corresponding receptor. So, if I take i th cluster, i th cluster represents the receptor of the i th radial basis function, so the situation is something like this.

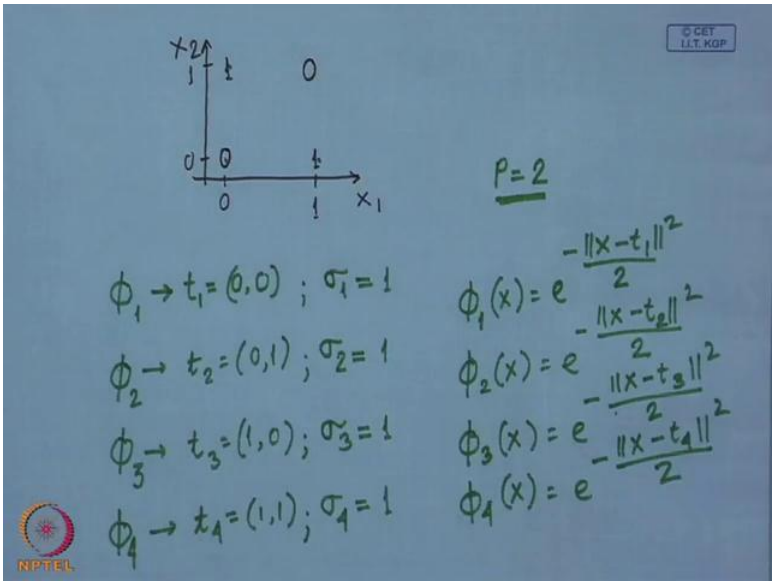
If I have a set of feature vectors, say these are the feature vectors belonging to different classes. Typically, what I do is I cluster this feature vectors into three different clusters. Every cluster center now represents a receptor, so this is 1 receptor, this is 1 receptor, this is 1 receptor. So, this is the receptor t 1, this is the receptor t 2 and this is receptor t 3. So, the first operation we have to perform is the cluster feature vectors and these clustering operations we will discuss in details in future lectures. Now, once I have these different receptors to find out what should be the spread of a particular radial basis function, what you do is force the i th receptor. I find out P number of nearest neighbors

or P number of nearest receptors and for this P number of nearest receptors, I compute what is the mean distance or root mean square distance.

So, there are different possibilities I can take any value I can choose any value out of these t number of receptor. So, what I do is the way I compute sigma i for the i th cluster of i th radial basis function is I have t i, which is the receptor for the i th radial basis function and then I take P number of nearest receptors, which are nearest to the t i. So, suppose 1 such receptor is t k, so what I do is I compute t i minus t k square, take summation of this for k is equal to 1 2 P as I have P number of receptors, 1 upon P of this and square root of this. So, this defines the spread of the i th radial basis function.

So, for every i th radial basis function, I have t i and have sigma i and once these 2 unknown then my radial basis function phi i of a x is simply e to the power minus x minus t i square upon 2 sigma i square. Now, let us see that by using this concept whether I can make a near classifier using the radial basis function concept for the x o problem and x o is very common problem, which is used for illustrating such operations.

(Refer Slide Time: 17:19)



So, as we have said earlier, if I take an x r function I have a 2-dimensional feature vector binary feature vector having components x 1 and x 2. Suppose, this represents 0, this is x 1 equal to 1 here. I have x 2 is equal to 0 and here I have x 2 equals to 1. The value of the x o function when it is 0, 0 is equal to 0 0 1 the value is 1. 1 0 the value is 1 and 1 1 in the value is equal to 0. So, find that here I have 2 dimensional binary feature vectors and

what I do is this 2 dimensional feature vector, I want to cost into a four dimensional space, by using four radial basis functions. So, I have the radial function, radial basis functions phi 1, phi 2, phi 3 and phi 4.

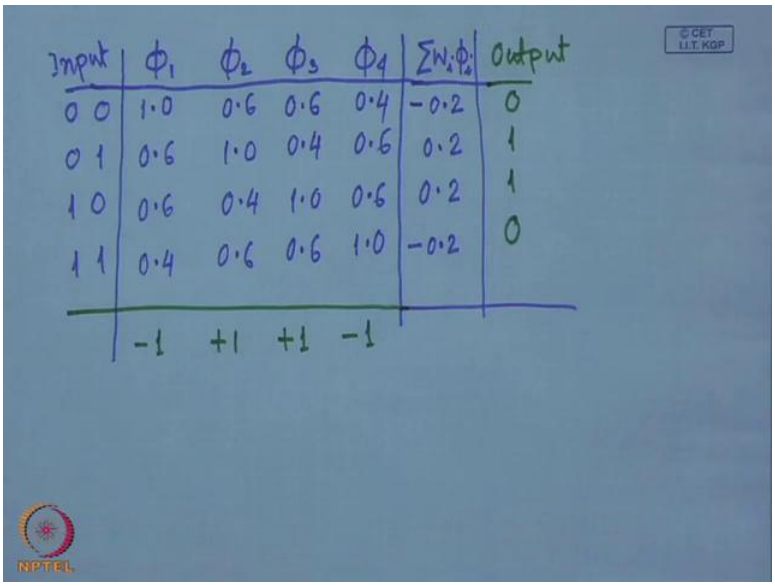
For phi 1, I choose t 1, is equal to that is the receptor of the radial basis function phi 1. Similarly, for phi 2, I can choose t 2 which is a 0 1, that is the basic receptor of the radial basis function phi 2. Similarly, the receptors of other radial basis functions I can choose as t 3 is equal to 0 1 and for this I choose t 4 is equal to 1 1. So, these are the four receptors for the fourth radial basis functions.

Next, I had to choose the spread sigma 1. For the first radial basis function I have to choose sigma 2 for the second radial basis function, sigma 3 for the third radial basis function and sigma 4 for the fourth radial basis function. Now, for this for every receptor I have to find out P number of nearest receptors and suppose I choose that the value of P is equal to 2. Now, here you find that for every receptor there are three neighbors, 2 of the neighbors are at a distance, are at distances of 1 and 1 of the neighbors is at a distance of 1.4, that is square root of 2. So, that is easily verifiable from here I have receptor over here, which is t 1, t 2 is at a distance 1, t 3 is at a distance 1, but t 4 is at a distance of square root of 2, which is 1.4 or 1.414.

So, when I take P is equal to 2, I had to take 2 nearest neighbors both of them are at distance 1 and root mean square distance of these 2 distances will also be equal to 1 so I have spread sigma 1 is equal to 1 I have spread sigma 2 also equal to 1, have spread sigma 3 also equal to 1 and have spread sigma 4 that also equal to 1. So, I get phi 1 x, which is of the form e to the power minus x minus t 1 square of this, upon 2 sigma on 1 square and sigma 1 being equal to 1. This will be equal to 2.

Similarly, for phi 2 x, I will have e to the power minus x minus t 2 square of this upon 2 phi 3 x will be e to the power minus x minus t 3 upon 2 and phi 4 x. So, if I compute to these values for each of the feature vectors taking 0 0 is 1 of the feature vector 0 1 as another feature vector 1 0 as another feature vector and 1 1 as another feature vector the functional values will be something like this.

(Refer Slide Time: 22:44)



So, I put that in the form of a table here I have input feature vectors inputs are 0 0, 0 1, 1 0 and 1 1 and I have the r p f functions phi 1 phi 2 phi 3 and phi 4. So, when you input the feature vector 0 0 to phi 1 you find that your x is equal to t 1. So, this exponent is equal to 0, which means that phi 1 x will be equal to 1. So, this phi 1 x over here this will be 1.0. Similarly, for phi 2 my x is 0 0 t 2 is 0 1, so if I compute this phi 2 x, you will find that this phi 2 x with be equal to 0.6.

Similarly, I just put the values over here. Phi 3 x will also be 0.6 and phi 4 x will be 0.4. When the input vector is 0 1, phi 1 x will be 0.6, phi 2 x will be 1.0, phi 3 x will be 0.4, phi 4 x will be 0.6. For 1 0 this is 0.6, this is 0.4, this is 1.0, this is 0.6 again and for the input feature vector 1 1, I have phi 1 is equal to 0.4, phi 2 x will be 0.6, phi 3 x will be 0.6 and phi 4 x that into 1.0.

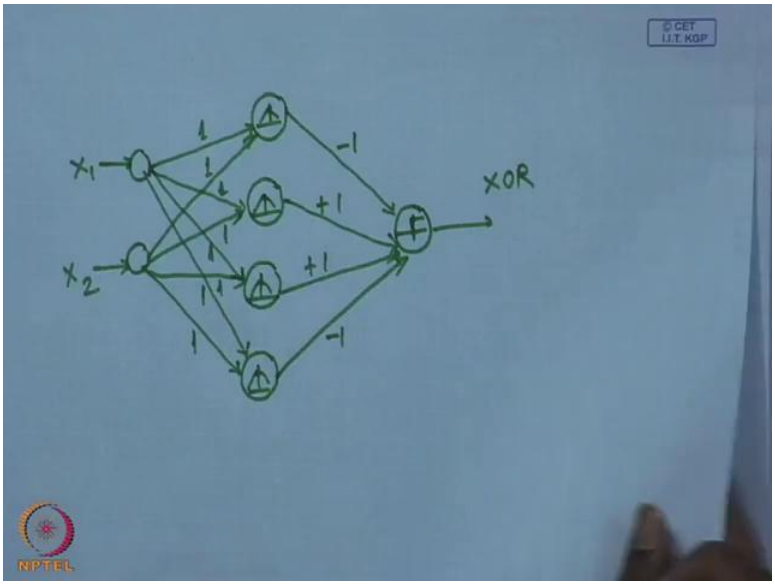
So, you find that given a 2 dimensional feature vector 0 0 this has been cost into a 4 dimensional feature vector where the components of this 4 but, dimensional feature vector are 1.0, 0.6, 0.6 and 0.4. Similarly, 0 1 is a 2-dimensional input feature vector, which has been cast into a 4 dimensional feature vector the components being 0.6, 1.0, 0.4 and 0.6. So, every input feature vector at the input feature vector is a 2-dimensional feature vector. Every 2-dimensional input feature vector is converted to a four dimensional feature vector by using four radial basis functions. Now, if a take a linear combination of this and for linear combination for phi 1 if I give an weight of, I give the

weight for phi 1, I give an weight of minus 1, for phi 2, I give an weight of plus 1 for phi 3, I give an weight of plus 1, for phi 4, I give an weight of minus 1.

So, function that I will finally, compute at the output at a node in the output layer will be phi 2 plus phi 3 minus phi 1 minus phi 4 and if I compute this, let us see what are the values that I get. So, here I will write, sum of W i times phi i where i varies from 1 to 4. So, here it will be 0.6 plus 0.6 is 1.2 minus 1.4, this will be minus 0.2. Similarly, here it will be 1.4 minus 1.2, so it will be plus 0.2. Here, again it will be 1.4 minus 1.2, so it will be 0.2 and here it will be a h1 0.2 minus 1.4, so this is minus 0.2.

And if I take a decision that if the value is more than 0 the output will be 1, if it is less than 0 without putting 0 then the final output that we have is here I write output. This will be 0, this will be 1 and this will be 0. So, which is nothing, but the extra function output. So, over here the architecture of the radial basis function that we have used is we had two input layer nodes.

(Refer Slide Time: 28:09)

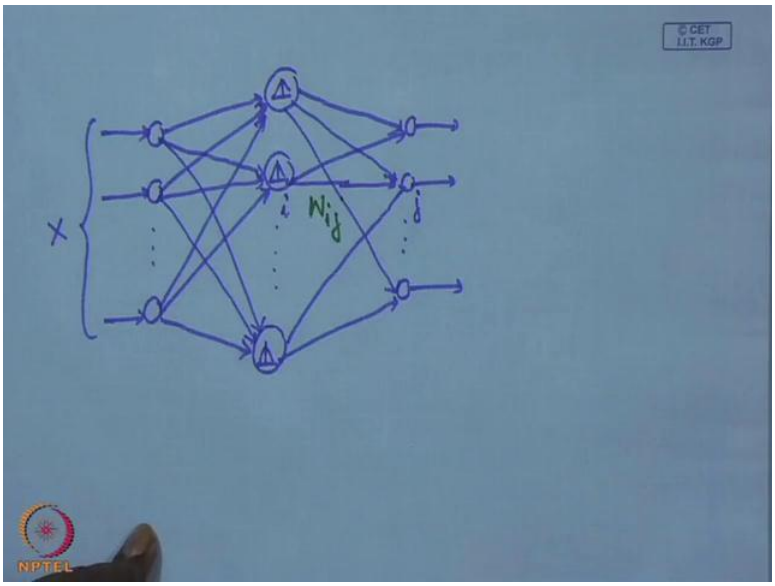


Where x 1 is stretch to 1 node and x 2 is paid to another node, I had 4 nodes in the hidden layer, which computes the radial basis function. And I had 1 node in the output layer which I can say that it is finding out it is a non-linear operator or a threshhold operator. The connections are like this, but each of this collection has a connection weight is equal to 1.

Over here these connections are as you can see over here phi 1 2 output layer node has a connection weight of minus 1, phi 2 2 output layer node has a connection weight of plus 1 phi 3 2 output node again has a connection wide of plus 1 phi 4 2 output layer node has a connection weight of minus 1. So, here the connections are minus 1, plus 1, plus 1, minus 1 and this output actually gives me the x o function, okay?

So, this example clearly shows that by casting the two-dimensional feature vectors into a four-dimensional feature vector. I can implement the x o function using a linear network or a single ((Refer Time 30:29)), because this part is nothing, but a single ((Refer Time 30:34)). Now, let us theoretically try to find out or try to find out an expression for the training of the output layer or how do I find out this connection. So, in general I have a network something like this.

(Refer Slide Time: 31:04)

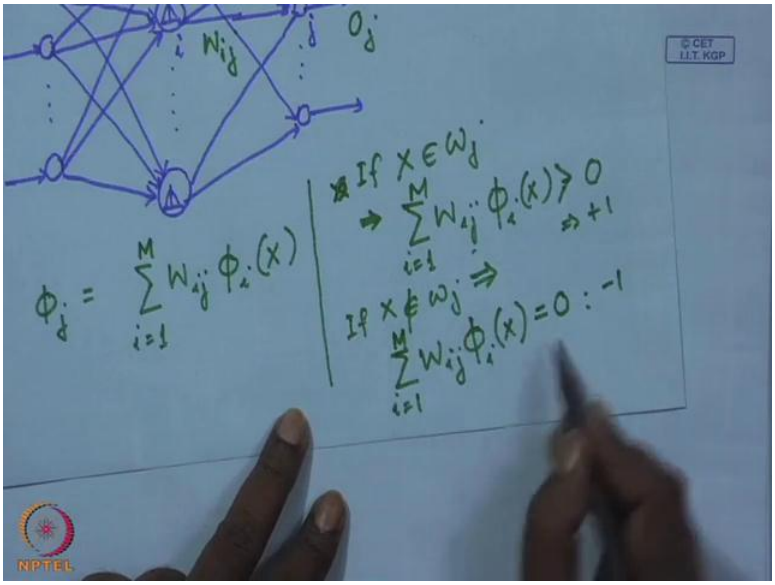


I have a set of input layers, set of input layer nodes, I have a set of hidden layer nodes and I have a set of output layer nodes. The feature vector is fed to the input layer nodes, so here I feed x, which are fed to the hidden layer nodes through connection weights, which are one and outputs of this hidden layer nodes. These are my radial basis functions. Outputs of the hidden layer nodes are connected to the output layer nodes.

So, like this and I take the output from entry output layer node. So, if my input feature vector x belongs to the i th class then output of the i th output layer node will have a high value likely to be 1 and outputs of all other output layer nodes will have a low value

likely to visit. I have shown that the i th node in the input layer is connected to the j th node of the output layer, through a connection weight say W i j.

(Refer Slide Time: 33:07)

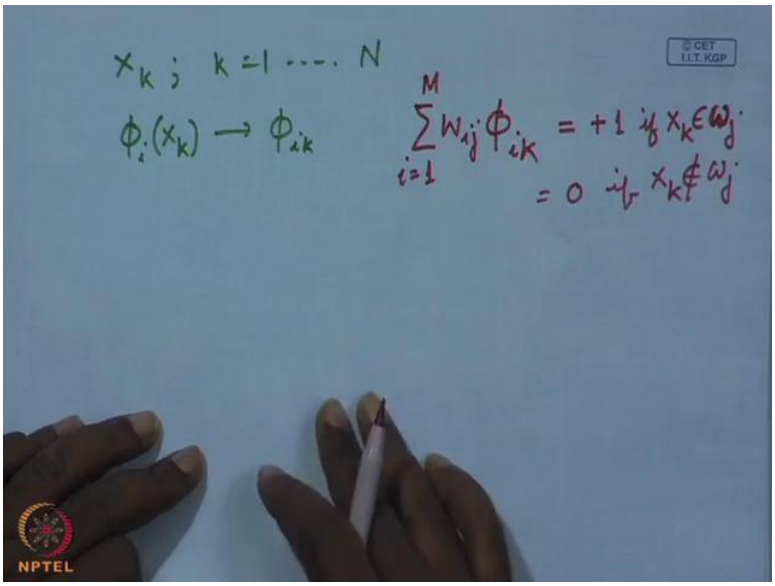


So, given this, if I say the output of the i th j th layer node is o j, I will have o j is equal to sum of W i j times phi i X for an input vector X and this summation I have to compute over all nodes in the hidden layer. So, here I will have this summation has to be computed over i is equal to 1 to M, as I have M number of nodes in the hidden layer. Naturally over here, if this feature vector X centre it if X belongs to plus omega j then I had to have sum of W i j times phi i X, i is equal to 1 2 M. This has to be equal, this must be greater than 0.

I will put this as plus 1 and if X does not belong to omega j that indicates that sum of W i j into phi i X, i varying from on to capital M that must be equal to 0 or I can also put it as minus 1. So, let us assume that if x belongs plus omega j W i j times phi I x that has equal to plus 1 and if x does not belong to omega j this has to be 0 and that is what have to be the output from the j th node in the output layer.

Now, taking this, now I can go for training of the output layer that means I have to find out what should be the values of this W I j. Now, if I compute only the connection weights, if I write. Now, can consider only the connection weights, which are connected to the j th node in the output layer then for every vector X k, suppose I have capital N number of vectors.

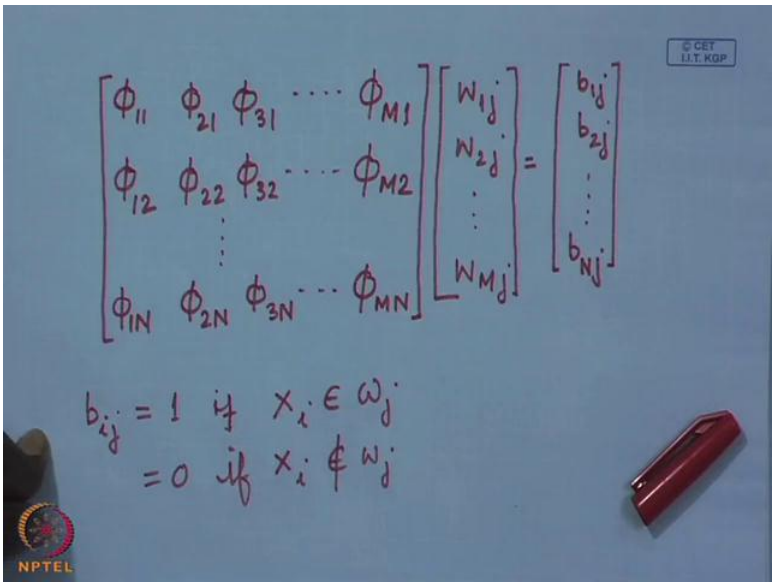
(Refer Slide Time: 35:58)



So, I have vectors X k for k value from 1 to capital N. I have capital N number of input vectors, which are given for any training purpose or for learning as ((Refer Time 36:13)) then phi i of X k. For simplicity I will write this as phi i k. now, by using this I can as I said that sum of phi I x k into W i j k

So, my condition is, if you remember this one, if you remember this one. So, sum of W i j into phi i k for i varying from 1 2 M. This has to be equal to plus 1 if X k belongs to plus omega j, if X k belongs to omega j and this has to be 0 if X k does not belong to omega j. So, this is the output that I expect, so for X k I have for every X k, I have such a kind of linear equation that this summation will be either plus 1 or 0 and all those capital in number of equations. Now, I can write in the form of matrix.

(Refer Slide Time: 38:04)

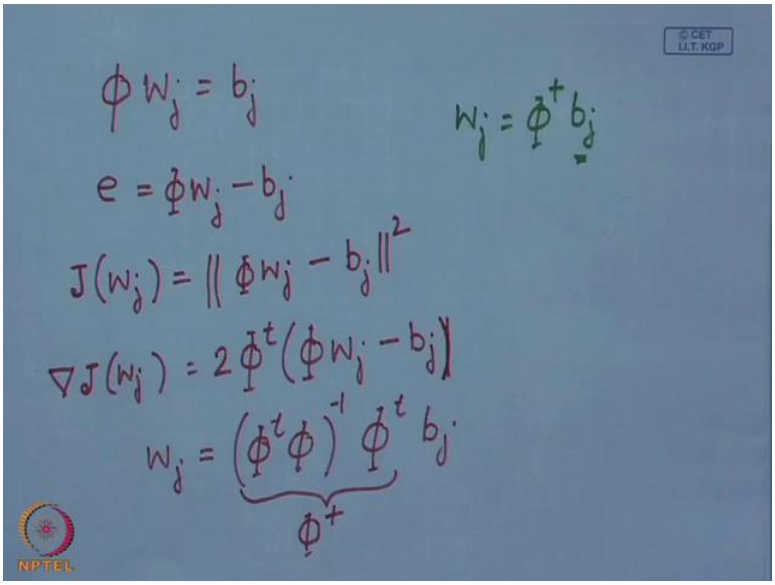


So, in the matrix form this can be written as, let me write the matrix equation that phi 1 1, which means phi 1 X 1, phi 2 1 that is phi 2 of X 1, phi 3 1, phi M 1, this means phi M of X 1. Similarly, phi 1 2, which means phi 1 of X 2, phi 2 2, phi 3 2 upto phi M 2 and as I have capital M number of samples for training so I will have phi 1 N, phi 2 N, phi 3 N upto phi M N, which indicates phi M of X N into W 1 j, W 2 j upto W M j.

So, you find that what it computes? W 1 j times phi 1 1 plus W 2 j times phi 2 1 continue like this W n j times phi M 1 that is for the first input vector x 1. Whatever is the output of individual middle layer nodes or hidden layer nodes. This equation simply makes a linear combination of outputs of the hidden layer nodes for the input feature 1 or X 1. So, this has to be equal to, again I put the output in the form vector b 1 j, b 2 j up b N j. For every b i j will be equal to 1.

If the corresponding X belongs to plus omega j and that will be equal to 0, if the corresponding X i does not belong to plus omega j. So, every b I j will assume a binary value either 0 or 1. So, this b i j will be equal to 1. If X i the corresponding input vector X i on belongs to plus omega j, the j th class or it will be equal to 0, if X i does not belong to omega j. So, this is the kind of situation that I have.

(Refer Slide Time: 41:43)



This four expressions, this matrix equation I can write in a short form, that is phi W j is equal to b j at this phi is this matrix, W j is weight vector which are connected to the output layer node j and b j is the output of the j th node in the output layer, which is represented in the small vector like this for different input vectors. So, if the network is properly trained, that is of all the W i j has got the trained value then this equation should be satisfied.

But, what we are trying to do is, we are trying to train the network that means we are trying to set the weights W j, so you cannot expect that this equation is satisfied initially. So, if this equation is not if this equality, this equality is not satisfied then what I can do is I can define an error e which is nothing but, phi W j minus b j. Now, training involves adaptation of this weight W j, so that this error can be minimized.

So, in order to do that as we have done one earlier for mean square error optimization for mean square error technique for classified learning or classified training. I can also define here function criteria function j of W j, which is given by phi W j minus b j log of this and then I take the gradient with respect to W j, so grad of j W j which will be simply 2 phi transpose into phi W j minus b j. By equating this to 0 what we get is w j is equal to phi transpose phi inverse of this into phi transpose b j.

As you have seen earlier this phi transpose phi inverse phi transpose phi inverse into phi transpose, this is what is called pseudo inverse and that is represented as phi plus. So, we

have W j by this pseudo inverse technique, so we have this W j is equal to phi pseudo inverse into b j, by b j is defined before hand every component of b j will be either 1 or 0.

It will be equal to 1 in the corresponding feature vector input feature vector belongs to plus omega j and the component will be equal to 0 if the corresponding input feature vector does not belong to plus omega j. So, I have this vector b j, phi actually indicates that what should be the output of the hidden layer nodes for every feature vector to from that I compute what is my matrix phi. So, once I have this matrix phi and I have this b j I can compute what will be the connection weights for different nodes in the hidden layer to the j th output layer node.

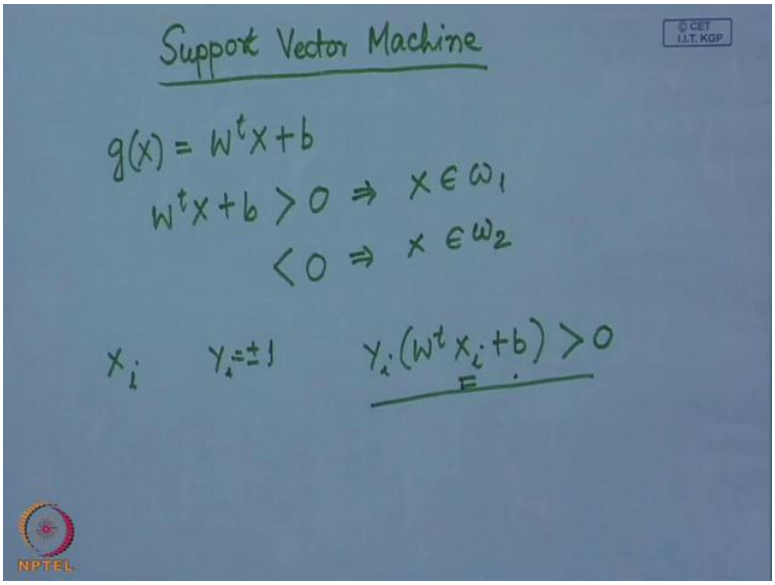
This if I do for every output layer node I can compute what is the connection weight for from different outputs of the hidden layer nodes to different output layer nodes and that is what completes my training of the r p f neutral network and the r p f neural network ready for classification. Now, if you compare this r p f neural network with your against some multilayer perceptron you will find that the training of the r p f neural network is faster than the training in multilayer perceptron, because in case of multilayer perceptron the training is d1 by back propagation algorithm which takes large number of iterations. So, the training of the r p f neural network will be faster than training of the multilayer perceptron.

The second advantage is that I can easily interpret, what is the meaning or what is the function of every node in the hidden layer, which is difficult in case of multilayer perceptron. I cannot easily interpret the role of different nodes in the hidden layer in case of multilayer perceptron. And not only that I also cannot easily decide that what should be the number of hidden layers and what should be the number of nodes in every hidden layer. So, those are the difficulties in case of multilayer perceptron, which is not there in case of r p f network.

However, r p f network has a disadvantage that though the training is faster, but you find that the classification takes more time. In case of r p f network than in case of m l p, because in case of r p f network every node in the hidden layer has to compute the radial basis functional value for the input feature vector, which is time consuming function. So, the classification in case, the classification in case of r p f network takes more time than the classification time in case of multilayer preceptor, okay? So, with this so we come to

a conclusion on the new radial basis function neural network. Now, over here I will just briefly discuss about another kind of classifier which is called a support vector machine.

(Refer Slide Time: 48:38)



So, I briefly discuss support vector machine. So, support vector machine is another type of linear classification. So, if you remember what we discussed in case of a linear classifier, that given a two class problem, we have said that I can define a discriminating function say g of X, which is of the form say W transpose X plus b. We have said in case of linear discriminator that if this g of X and W transpose X plus b this is greater than 0 that indicates that feature vector X belongs to plus omega 1. If this is less than 0 then feature vector X belongs to plus omega 2.

So, here we find that for classification purposes the actual value of g X is not really very important, but what is important is what is the sign of g X. If the sign is positive I infer that x belongs to plus omega 1 if the sign is negative I infer that x belongs to plus omega 2, right? So, over here with every X, if I or if with every x I indicate a number say y I that y I can be either plus 1 or minus 1.

In that case this Y I times W transpose X i plus b it will always be greater than 0. If the sample X i is properly classified, which is quite obvious, because if I say that Y i equal to plus 1 for a sample X which belongs to class omega 1 and for a sample which belongs to class omega 1 this W transpose X i is greater than 0 Y i is also positive. So, Y i times this will obviously be greater than 0 if X i belongs to class omega 2 then W transpose X i

will be less than 0. For that I have set Y i equal to minus 1 times W transpose X i plus b will obviously be greater than 0.

This is a concept that actually we have used when we have discussed about the perceptron criteria or designing the linear classifier that is for every feature vector belonging to class omega 2. We have negated the feature vector before we try to design the classifier, so that for every feature vector irrespective of whether the feature vector belongs to class omega 1 or the feature vector belongs to class omega 2 my discriminate function value will always be positive.

If the feature vector is correctly classified, so that is to if the feature vector belongs to class omega 1 or even if the feature vector belongs to class omega 2, because of the feature vectors belonging to class omega 2 before trying to design the classifier we have negated this feature vector. So, we will discuss about the support vector machine more details in our next class.

Thank you.