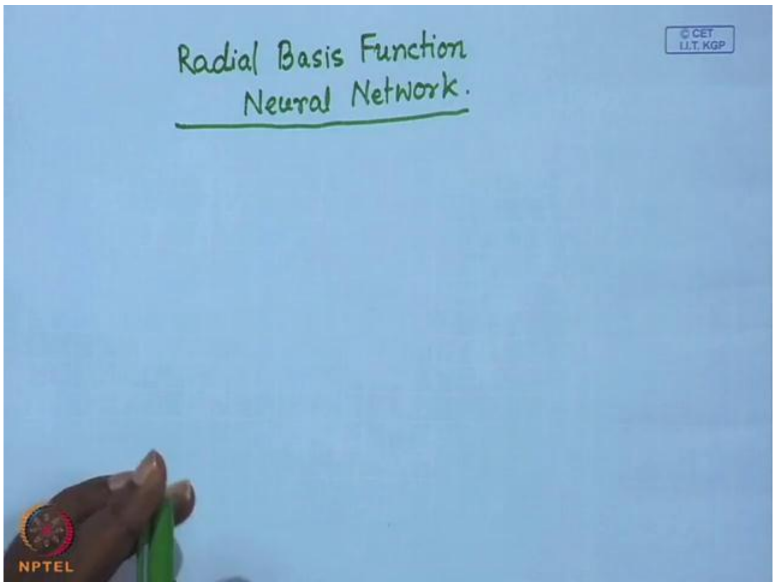
Pattern Recognition and Applications

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

Lecture - 27
RBF Neural Network

(Refer Slide Time: 00:22)

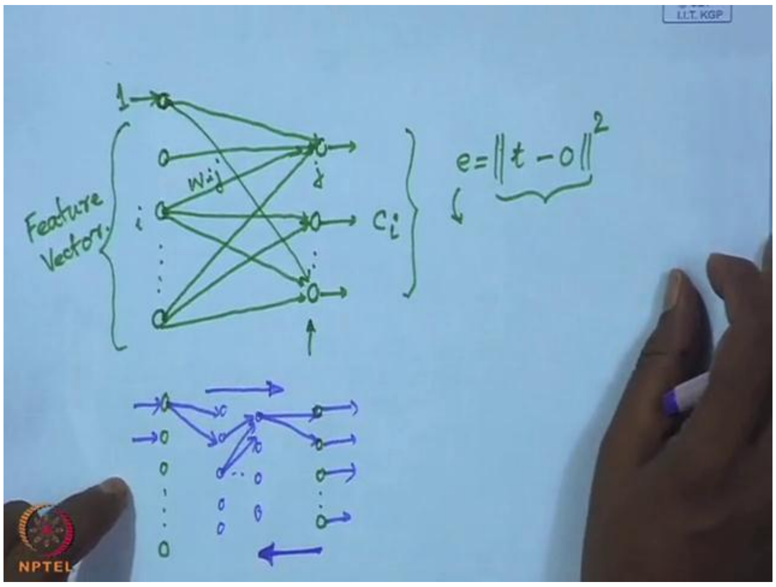


Radial Basis Function Neural Network.

Hello. So, in the last few classes we have talked about the neural networks and the application of neural networks for patterned recognition. Particularly, we have talked about two types of neural networks; one was a of the Hopfield neural network, as we have said that Hopfield neural networks are basically meant for binary feature vectors. So, if I have a number of binary feature vectors the, Hopfield neural network basically wants or remembers those feature vectors by updation of weights. So, learning process is basically updation of weights and in case of back propagation neural network, the learning is also by means of updation of weights.

But for back propagation or feed forward neural network, the neural network is not confined to the binary feature vectors, only it considers the feature vectors which contend ideal components also. Now, when we talked about the feed forward neural network or back formation neural network, we have said that it is a multi-layered neural network, which has got one layer, which is input layer and it is just a buffer layer, whose responsibility is to feed, is to just pass the information to the layers above it. And in the simplest case of the back propagation neural network, we had single layered perceptron, which consists of only one output layer, in addition to the input layer. So, the neurons in the output layer so if I just draw it in case of a single layer perceptron, the network architecture of single layer perceptron was something like this.

(Refer Slide Time: 02:25)



We had an input layer and we had an output layer so the number of nodes in the input layer is same as, the dimensionality of the feature vector or if the feature vector is of the dimension d then the number of nodes in the input layer is d plus 1. So, one node is to take care of the bias term and the number of nodes in the output layer is same as, the number of classes. So, the network architecture was something like this, from each input layer node we had a connection to every output layer node so it was something like this and so on. So, a feature vector is straight to the input layer nodes and one of the node was kept at a constant value, which is equal to 1.

That is meant for filling the bias term and outputs of this different nodes in the output layer, they indicated the corresponding classes Ci. Every node in the input layer is connected to every other node in the output layer, through a connection weight say wij. So, ith node in the input layer is connected to the jth node in the output layer, through a connection with wij. So, during training what we had done is, we had failed the feature vectors for which the classes are known. And if the class of a feature vectors known, that means I know that what should be the output corresponding to that particular feature vector.

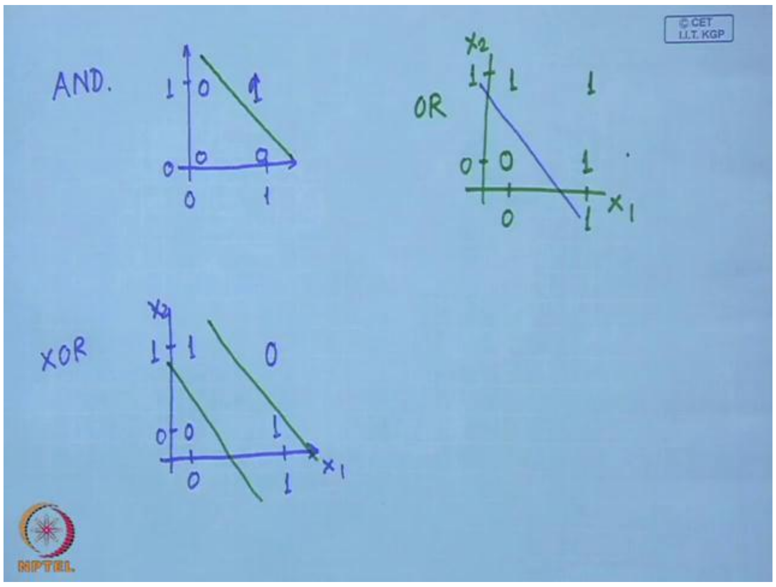
So, that output is an output vector say t and when the network is not properly learnt, in that case for an input vector, I get an output which may not be equal to t. So, I get an output which is o or as t is the target output. So, I have an error e which is equal to square of this. So, during training of this back propagation neural network what we had done is, we tried to minimise this error e by means of updating this connection weights. So, this was expressed as a function of a set of connection weights wij and then to gradient descent procedure. We have reduced the error by updation of the connection weights wij.

In case of multilayer perceptron, we had the same input layer, input layer is always possible, and we also had the number of nodes in the output layer, which is same as the number of classes. In addition, we had one or more hidden layers however, the connection the pattern remains the same, and every node in the input layer is connected to every other node in the layer above it. So, it continued this way and so on and finally, we get the output. However, the training for this back propagation neural network was same as, a single layer perceptron that is we, for unknown feature vector which is input to this neural network.

We know what the output is or we know what is the target output. And when the neural network is not properly learnt we get an output for the same vector, same input feature vector, which is not same as the target output. So, there again we get an error and this error when I express in the form of a function, of the weight matrix, of the connection weights then will try to reduce the error by optimising or by adopting the weights. And when we modified the weights, when we adapt the weights the error information for reduction of the error, that propagates from output layer to input layer so it to propagate in the reverse direction.

So, it is like this, the error information or the weight updation information that propagates from the output layer to the input layer. So, that is why it is got back to perdition land, but during the classification, the information moves from the input layer to the output layer. So, the information is paved from input to output so that is why it is feed forward network whereas, the learning algorithm is called as a back propagation learning algorithm. Now when we discussed about this, feed forward network or back to propagation network, we have said that if the patterns or if the classes are linearly separable then a single layer perceptron is sufficient. But if the classes are not linearly separable then single layer perceptron is not sufficient, we have to have one or more hidden layers. And what these hidden layers do is, it represents a non-linear boundary by a set of piecewise linear boundaries and to explain this, we have taken few examples.

(Refer Slide Time: 08:53)



We have taken an AND function. So, in case of AND function if I represent the AND function as, a two-dimensional binary vector. So, I have the input components 0 0 0 1 1 0 and 1 1 so if the function is an AND function, only when the input is 1 1, the output is 1. In all other cases 1 0 0 0 or 0 1 output is 0 and now, if this AND function we consider to be a classification problem then you find that it is quite obvious. I can draw a straight line separating the set of 0's and the set of 1's, so that means the input vector 1 1 that is putting one class. And all other combinations of the input components, that is 0 0 0 1 1 0 that is, those are put in another class.

And I can separate these two classes by a straight line so that means that this particular function, AND function is a linearly separable function. Similarly, if I take an OR function here again I put it as x1 x2 somewhere here, let us put 0 1 0 1 so only in case of 0 0 both x1 and x2 components, when they are 0 the output is 0. In all other cases the output is 1 so I have this type of situation and again here, we find that I can separate the

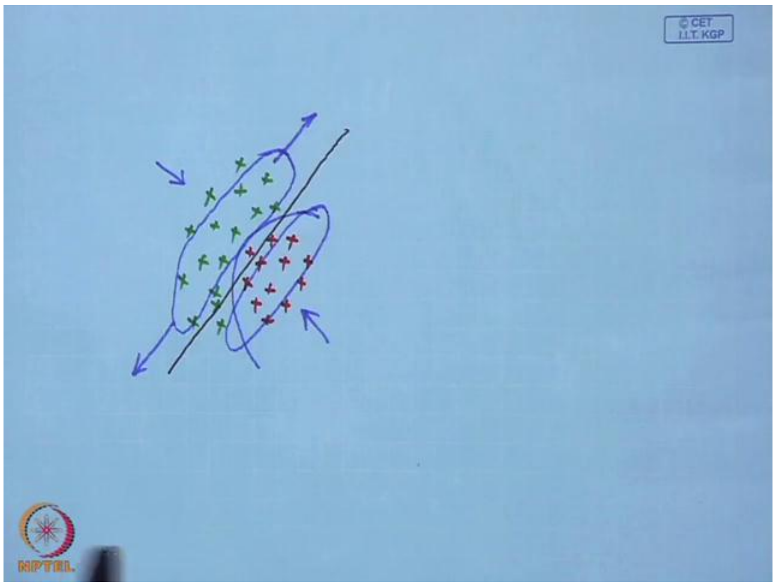
1’s and the 0 by a straight line. So, which clearly indicates that, this is also a linearly separable problem, but we have faced difficulty in case of XOR function.

So, in case of XOR function if I put it in the same fashion, I have x1 x2 0 1 0 1 so when the input is 0 0 the output is 0, when the input is one 1 1, output is also 0, when the input is 0 1 or the input is 1 0 the output is 1. Now given this type of situation, you find that I can no longer separate the outputs 0’s and 1’s by single straight line. Rather, I need to have two different straight line over here, to separate the 0’s and 1’s. So, this clearly says that the XOR function is not linearly separable. And when we discussed about this back propagation neural network, we have said, we have shown the AND function and OR function they can be represented, if they can be implemented using single layer perceptron’s.

But when you go for XOR function, the XOR function cannot be implemented by single layer perceptron rather, we need at least one hidden layer for implementation of XOR function by a feed forward neural network because it is not linearly separable. So, now, today will discuss about another kind of network which is called radial basis function network. So, what radial basis function network does is, it performs a non-linear transformation over the input vectors, before the input vectors are fed for classification. So, by using such non-linear transformation, it is possible to convert a linearly non separable problem to a linearly separable problem.

It is also possible what this, RBF network or radial basis function network does is, it increases the dimensionality of the feature vectors. So, I will come to that later first let us see that, how this introduction of non-linear function to the input feature vectors, before we go for classification, can make a linearly known separable problem, can convert a linearly non separable problem to a linearly separable problem. So, let us just take a situation something like this.

(Refer Slide Time: 14:09)



Say, I have a set of feature vectors, which are at this form. So, here you will find that, if I say that all these green crosses represent feature vectors belonging to one class and all the red crosses represent feature vectors belonging to one class, another class. So, clearly you will find that, these two feature vectors cannot be separated by straight line rather, what I need to have is, I have to have a curve, a quantity curve at least to separate these two classes. Now if I put, if I impose on non-linear transformation something like this, that a transformation which will expand the feature vectors in this direction and contract the feature vectors in this direction.

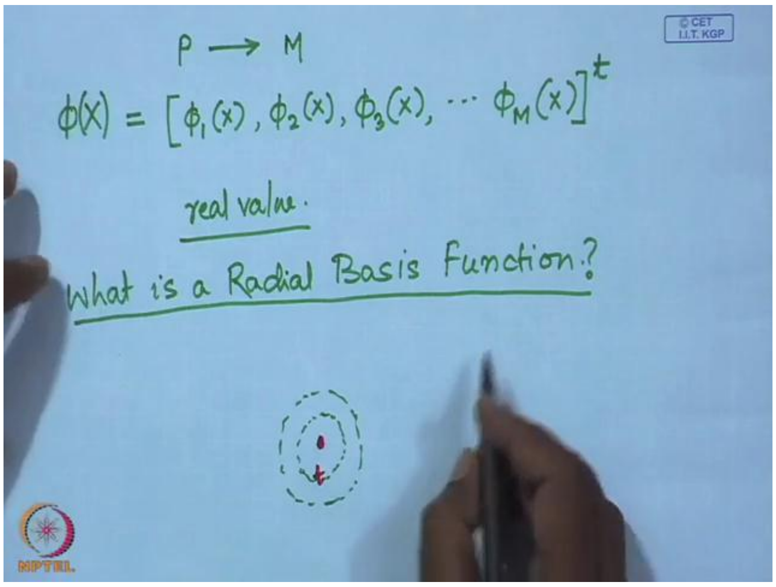
So, if I do this these feature vectors will be converted, will be clustered into a cluster something like this and these feature vectors, it is possible they will be clustered into cluster like this. And because of this non-linear transformation, now it may be possible that, I will be able to find out a straight line, which separates these feature vectors from these feature vector. So, that is why I say that, if I use a non-linear transformation, if I impose a non-linear transformation to the feature vectors. Before I put them I go for classification purposes then it may be possible to that a non-linearly separable problem can be converted to a linearly separable problem.

The other one as I said that the RBF network, the radial basis function network also increases the dimensionality of the feature vector. Because it has been found that, if we increase the dimensionality of the feature vectors then classification problem which is linearly non separable in ((Refer Time: 16:53)) dimensional space. It may be possible that, they will be linearly separable in higher dimensional space. And more and more you increase the dimension, the linear seperability, the possibility of linear separablity increases. So, it is something like this, if I put it in this forms of the tips of my fingers, they represent different feature vectors.

And the tips of the fingers of the left hand, they represent feature vectors belonging to one class and the tips of the fingers of the right hand, they represent feature vectors of another class. And these feature vectors they are coplanar so all of them are lying in the same plane. So, if the situation is like this obviously, this set of feature vectors are not linearly separable.

However, if I introduce a third dimension something like this, I push ((Refer Time: 17:51)) the forward and the other one back ward. And in this third dimension, now we find that they are linearly separable so this is just a simple logic that, if I increase the dimensionality. Then it is possible to increase the possibility of linear separablity, among the classes which are linearly non separable in lower dimensional space. So, the RBF network actually imposes both, it imposes a non-linear function, a non-linear transformation to the feature vectors. And it also increases the dimensionality of the feature vectors.

(Refer Slide Time: 18:48)



$P \longrightarrow M$

$\phi(X) = [\phi_1(x), \phi_2(x), \phi_3(x), \cdots \phi_M(x)]^t$

real value.

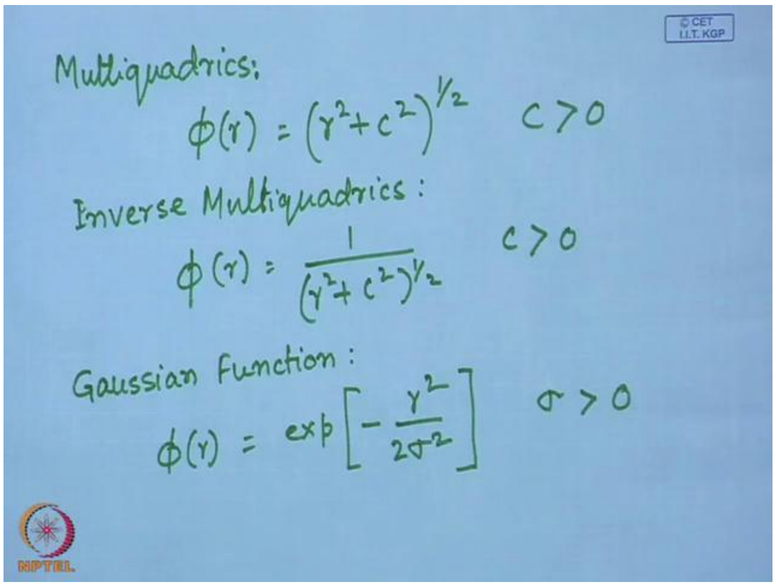
What is a Radial Basis Function?

So, if I have set of feature vectors whose original dimension is the dimension P. I increase the dimension of the feature vectors to dimension M. So, how it is to be done, a given feature vector x is subjected to function phi. So, phi of x, and this phi of x is the form of phi 1 x, phi 2 x, phi 3 x it is up to phi M x. If the original feature vector a x is of dimension P, for each such feature vector x, I impose this state of phi functions, n number phi functions. And each of these phi functions produce a real value so as I have M number of phi functions so by application of this phi function onto this feature vector x.

I create a number of real components so what I get is an M dimensional feature vector and if M is greater than P then obviously, I am increasing the dimensionality of the feature vector from P to M. And in case of radial basis function, each of these phi function, whether it is phi 1, phi 2, phi 3 up to phi M each of this phi functions are radial basis functions. Then what is a radial basis function? So, let us try to see what a radial basis function. Every radial basis function has a receptor, the radial basis function has a receptor say t and as I move radially away from this receptor, the value of the function goes on increasing or goes on decreasing.

And the value at a point, depends upon the radial distance of the point from the receptor t. Value of the function is either maximum or minimum at location t so if I put concentric circles around this receptor t, the value of the function phi on these each of this concentric circles are constant. So, there are different choices of radial basis functions.

(Refer Slide Time: 22:38)



Multiquadrics:

$$\phi(r) = (r^2+c^2)^{1/2} \quad c>0$$

Inverse Multiquadrics:

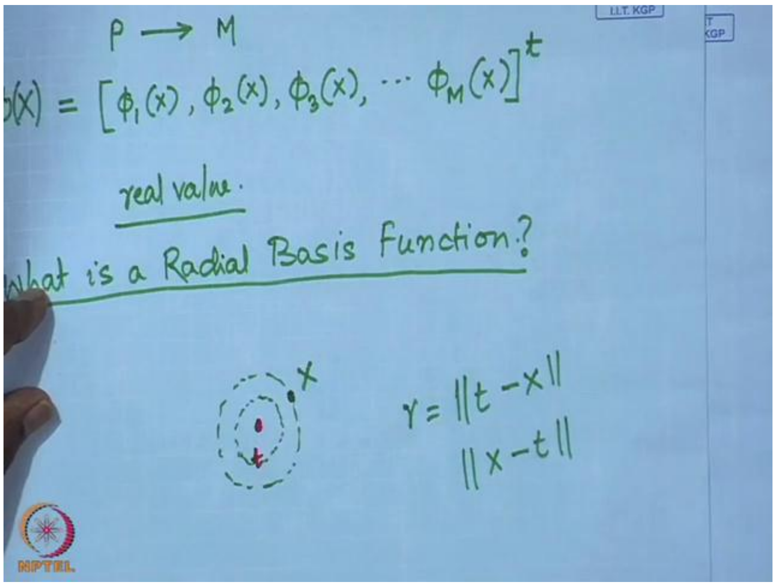
$$\phi(r) = \frac{1}{(r^2+c^2)^{1/2}} \quad c>0$$

Gaussian Function:

$$\phi(r) = \exp\left[-\frac{r^2}{2\sigma^2}\right] \quad \sigma>0$$

I can put those radial basis functions as, one of them is called multiquadrics, which is given by phi of r is equal to r square plus c square to the power half for c greater than 0. Then I can have inverse multiquadrics, are the function phi r is given as, just inverse of quadrics that is r square pulse c square to the power half for c greater than 0. I can also have used Gaussian functions, where phi of r is given as exponential minus r square upon 2 sigma square for sigma greater than 0. And this r is a measure of the distance of the point from, what I said as the receptor of the radial basis. So, this r if I want to compute this radial basis function at a point say x, coming over here.

(Refer Slide Time: 24:24)



$P \longrightarrow M$

$$\phi(x) = [\phi_1(x), \phi_2(x), \phi_3(x), \cdots \phi_M(x)]^t$$

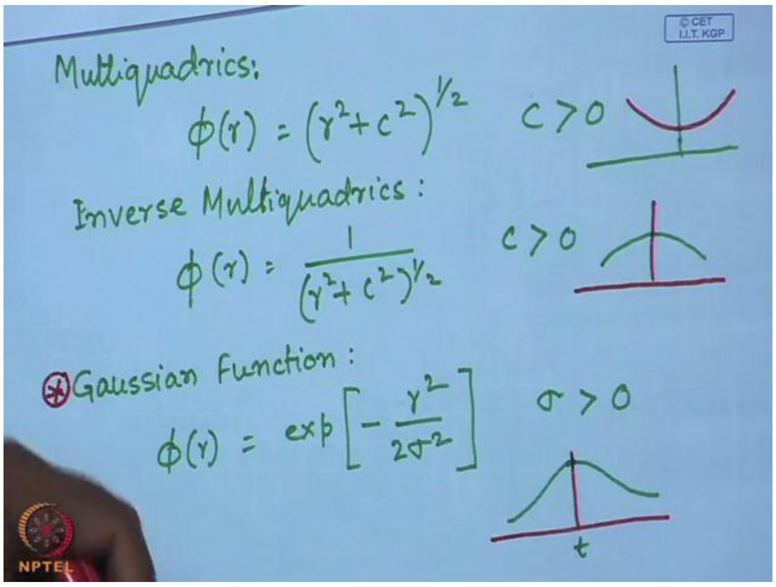
real value.

What is a Radial Basis Function?

$$r = \|t-x\|$$
$$\|x-t\|$$

If I want to compute, the radial basis function at a point x, t is the receptor of this radial basis function. Then r is nothing but t minus x or x minus t which is nothing but the distance between x and t. So, you will find that in all these different cases if I use this multiquadrics then at r equal to 0 the value of the radial basis function is equal to c, which is the minimum. And as far r goes on increasing, the value of the radial basis function goes on increasing.

(Refer Slide Time: 25:16)



Multiquadrics:

$$\phi(r) = (r^2 + c^2)^{1/2} \qquad c > 0$$

Inverse Multiquadrics:

$$\phi(r) = \frac{1}{(r^2 + c^2)^{1/2}} \qquad c > 0$$

Gaussian Function:

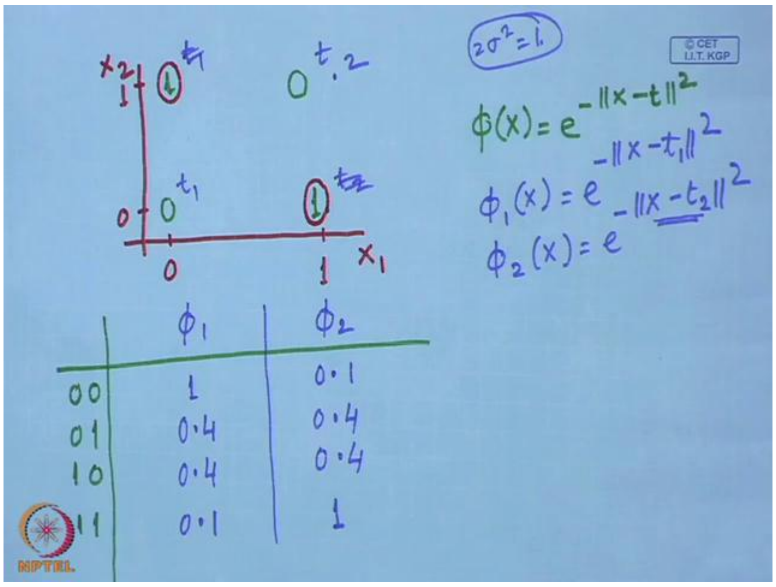
$$\phi(r) = \exp\left[-\frac{r^2}{2\sigma^2}\right] \qquad \sigma > 0$$

t
© CET I.I.T. KGP

So, the nature of the radial basis function for multiquadrics will be like this, when I go for the inverse multiquadrics, at r equal to 0, that means when the point x and the point t that coincide. That is at the receptor point, the value of the inverse multiquadrics is minimum which is nothing but 1 upon c, this is the maximum which is nothing but 1 upon c. And as r goes on increasing, the value of radial basis function goes on decreasing so the kind of, the nature of the function with inverse multiquadrics will be like this. Just inverse of multiquadrics at, in case of a Gaussian function you all know that, this has a very very common form, the Gaussian function, will be something like this.

So, at the receptor location t, the value of the function phi will be maximum, and as you move away from the receptor point, the value of the function phi r that goes on reducing following this pattern. And it is this Gaussian function, which is most commonly used in case of radial basis function. Now, let us say that how by using this radial basis function neural network, we can convert a linearly non separable problem to a linearly separable problem, even keeping the dimensionality of the feature vector same. And so for doing that I will use the same XOR problem.

(Refer Slide Time: 27:23)



$2\sigma^2 = 1$

$$\phi(x) = e^{-\|x-t\|^2}$$

$$\phi_1(x) = e^{-\|x-t_1\|^2}$$

$$\phi_2(x) = e^{-\|x-t_2\|^2}$$

	ϕ_1	ϕ_2
00	1	0.1
01	0.4	0.4
10	0.4	0.4
11	0.1	1

So, I have this x2, x2 here I have 0, here I have 1 again 0, here it is 1, I have 0 over here.. So, these are the outputs of an XOR function, what I will do is, I will use a radial basis function to nonlinearly transform these feature vectors into another space. And the radial basis function that I use is, a Gaussian function so the form of the radial basis function will be phi x, which will be given by, e to the power minus x minus t square and this t is the receptor and as I said that, I will not increase the dimensionality. So, I will use two of the receptors, one of the receptor I consider is this point at location 0 1 and other receptor I consider this point at location 0.

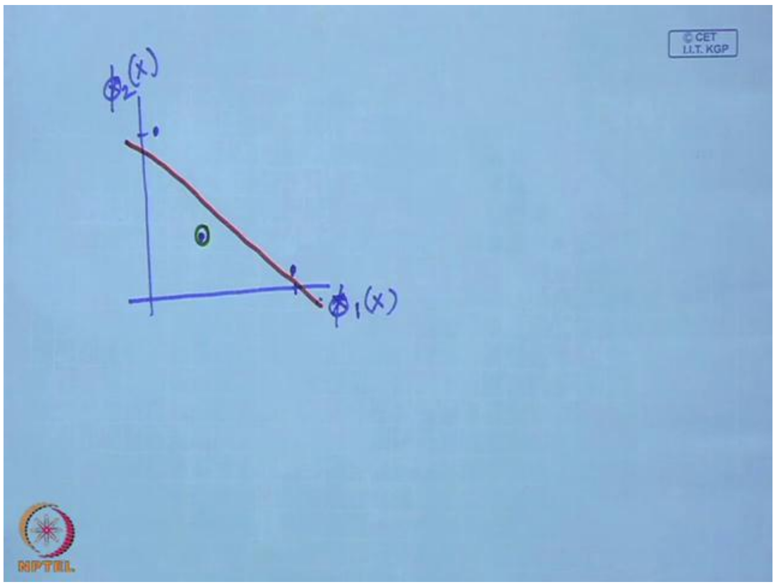
So, by using this if I compute the non-linear transformation then the values that I will have will be something like this. I use this non-linear transformation so I will put this inputs my inputs of 0 0 0 1 1 0 and 1 1 and I will have the output functions. One is phi 1 and the other one is phi 2 so if I say that this is my t 1, the receptor t 1 and this is my receptor t 2. So, phi 1 x will be simply e to the power minus x minus t 1 square and phi 2 x will be e to the power minus x minus t 2 square.

So, here you will find that, I assume that 2 sigma square that is equal to 1 however, that does not matter for this non-linear transformation. So, define that when it is 0 0 then what will be my phi 1 of 0 0, phi 1 of 0 0 will be simply equal to, let me use the receptor like this. Let this be t 1 and let this be t 2 so phi on1e of 0 0 will be equal to 1 and if you compute this term phi 2 of 0 0 will be 0.1. Phi 1 of 0 1 will be equal to 0.4 and phi 2 of 0 1 will also be equal to 0.4, phi 1 of 1 0 will be 0.4, this will also be 0.4. And for 11 phi of 1 will be 0.1 and phi 2 will be equal to 1. So, this is the kind of situation that I have so what have used is this 0 0 has been used as one of the receptors of the radial basis function and 1 1 has been used as another receptor of the second radial basis function.

So, obviously 0 0 as it coincides with my receptor t 1, so I have the situation that x minus t 1 square, that will be equal to 0 so this is e to the power of 0, which is equal to 1.So, phi 1 of this x which is 0 1, 0 0 which is equal to 1 similarly, if I compute phi 2 of x that will be equal to 0.1. For 0 1 phi 1 of the x will be 0.4, phi 2 of x will also be 0.4, for 1 0 which is this point, phi 1 of x will be equal to 0.4, phi 2 of x will also be equal to 0.4.

Because of these x 1 initial functions, for 1 1 if x is equal to 1 1 this coincides with t 2 so phi 2 of x this distance be equal to 0 now, phi 2 of x will be equal to 1 and phi 1 of x, this is the receptor the phi 1 of x will be equal to 0.1. So, you find that 0 0 has been mapped to 1, 0.1, 0 1 have been mapped 0.4, 0.4, 1 0 has been mapped to 0.4, 0.4, 1 1 has been mapped to 0.1, 1.

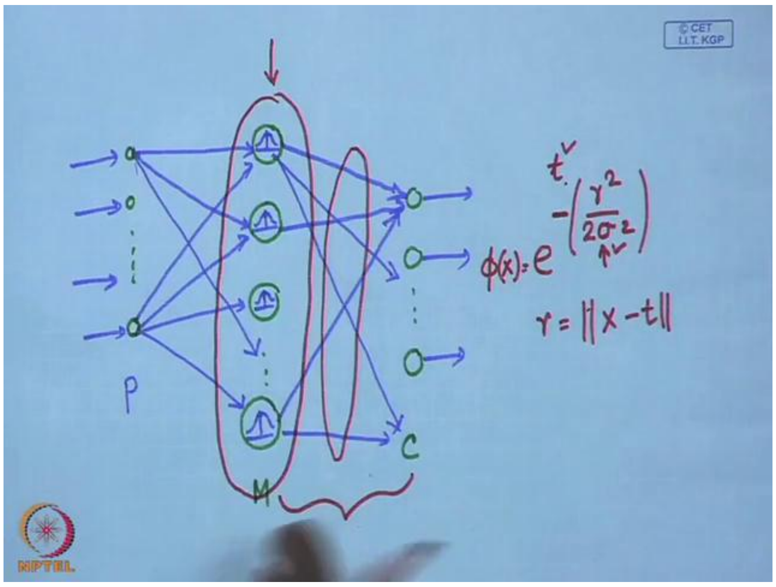
(Refer Slide Time: 33:48)



So, if I plot this, the kind of plot that I have is 0 0 is now mapped to 1, 0.1 which will be somewhere over here, this is my x1, this is my x2. Now, this will be phi 1 instead of x1it will be phi 1 and this will be phi 2. So, this is phi 1 of x this is phi 2 of x so 0 0 has been mapped to 1, 0.1, 0 1 has been mapped to 0.4, 0.4 which will be somewhat here. 1 0 that is also mapped to 0.4, 0.4 it is the same point so both the points up to be, now has been mapped to the same point in phi 1 phi 2 domain. And that has happened because of the non-linear transformation whereas, 1 1 has been mapped to 0.1, 1 somewhere over here so this is 1.

This being the situation now you find that, I can draw a straight line separating this points say, problem which was nonlinearly, which was not linearly separable in original space, it has now been linearly separable in 5 steps, simply by using: non-linear transformation. And the possibility of the linearly separablity, between the classes increases as we go on increasing, the number of, the dimension of the feature vectors, by using these five functions. So, by using this now the architecture of the network will be of this form.

(Refer Slide Time: 35:49)



We have the input layer as before, it always has been present because this is the layer which accents the input vectors and feeds to the next higher layers. We will have only one hidden layer and the number of nodes in the hidden layer will be equal to M, where M is the dimension, to which we want to increase the dimensionality to the feature vectors. So, if I have P number, P is the dimensional the input feature vectors, we have P number of nodes at the input layer and if M is the increased dimensionality then M is the number of nodes in the hidden layer. Each node in the hidden layer, actually performs a radial basis function and as I said, it is something like this, that is each node represents or computes a radial basis.

And in the output layer, we will have number of nodes which is same as the number of classes c, now from each node in the input layer I have a connection to each node in the hidden layer. And from each node in the hidden layer, I have a connection to each node in the output layer. Finally, I have outputs from the classifying neurons and here I freed the input feature vectors so this is, what is the architecture? So, the nodes in the hidden layer, they actually perform the radial basis function and nodes in the output layer, they perform linear combination of the outputs of the hidden layer nodes. So, here I will have a linear classifier so actually classification is done only at the output layer.

Whereas, at the hidden layer only we perform the radial basis function so that is a transformation of the input feature vector to a hidden space. Because this is my hidden layer so you can say that it is a transformation to a hidden space, so when I have a number of nodes in the hidden layer which performs radial basis function. So, I have two levels of training, the first level of training is, training of the hidden layer nodes. So, what does this training mean, as every node in the hidden layer is a radial basis function so for each of the radial basis function I have to have a receptor t. So, one of the purpose of training of the hidden layer nodes is, to find out that what should be this receptor, t.

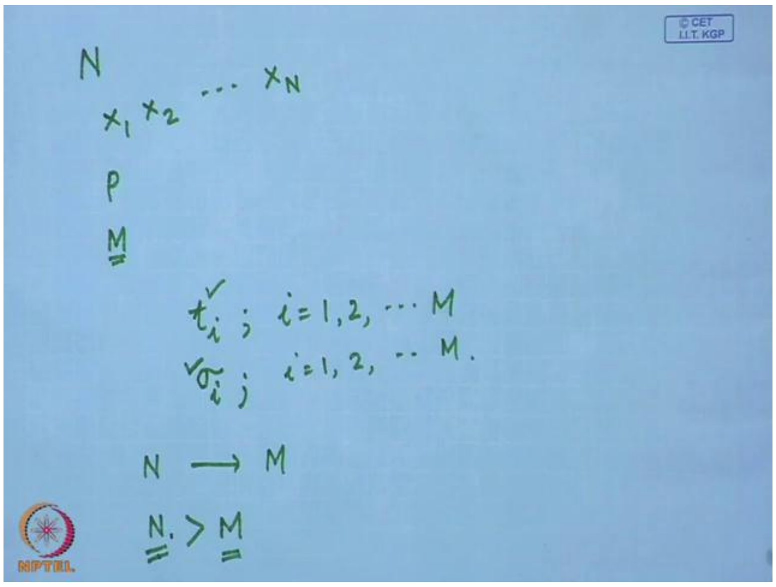
And then if I use the Gaussian function then we have saved that, the radial basis function is of the form e to the power minus r square upon 2 sigma square so that is what is my phi of x, where this r is nothing but the distance between x and t. So, this sigma actually indicates that what is the spread of this radial basis function, so that is also what is to be trained. So, by saying that I want to train the hidden layer nodes, for each of the nodes I have to find out what is the t and for each of the nodes I have to find out what is the sigma.

So, that is what is meant by training of the hidden layer nodes and at the second level I have another training operation, by which I have to train these weight vectors, which connects the outputs of the hidden layer nodes to, the output layer nodes because it is the linear combination of the outputs layer nodes of the hidden the nodes, which decides that to which class a sample will belong. And that will be outputted, that with the available at the output of the, output layer nodes. So, let us see that, how you can train a node in the hidden layer so by saying that the hidden layer nodes are to be trained, what, let us see the situation that we have. If I go for supervised classification or supervised learning what I have is, I have set of feature vectors or which a level with it's ((Refer Time: 42:01)) belongingness, which are to be used for training the neural network.

In case of multi layered perceptron, in case of single layered perceptron we did not have any non-linear transformation so from the level of the input feature vector and actual level that you got, at the output of the output layer nodes, we have defined an error function. And our aim was that by updation of the connection weights, we have to reduce that error function or we have to minimise that error function. So, the algorithm was that in order to minimise the error function, how the connection weights are to be modified.

And that modification of the connection weights was carried out from the output layer to the input layer, in the backward ways. Or as in the case of RBF network, the training consists of two parts, the first one is training of the hidden layer nodes. And when I say it is the training of the hidden the nodes that means, it is for every hidden layer which represents a radial basis function, I have to find out that what should be the receptor. And also I had to find out what is the spread of the radial function, of the radial basis function which is sigma. So, if we are given on number of training vectors.

(Refer Slide Time: 43:45)



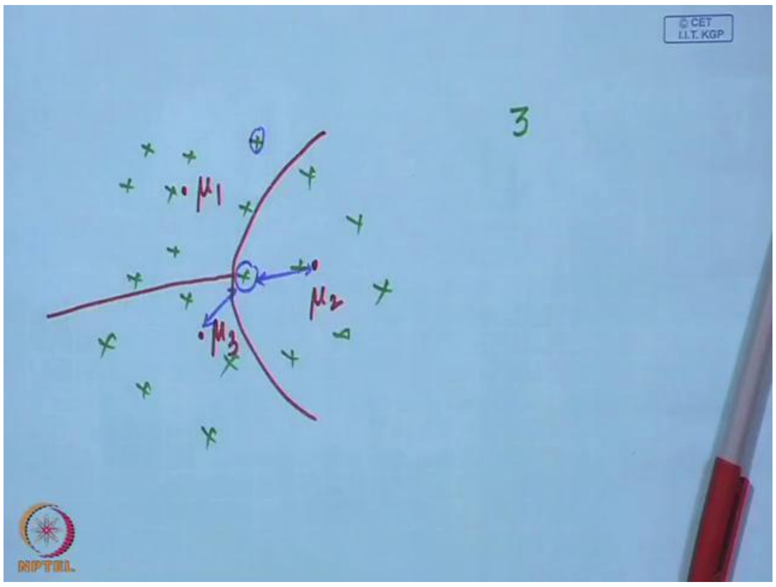
So, we are given N number of training vectors, running from say x1, x2 up to say xN, we have N number of training vectors. Each of these vectors is of dimension P and I want to transform these vectors to a vector of dimension capital M. So, naturally at the hidden layer of the node, at the hidden layer I have to have capital M number of nodes. And when I have this M number of nodes so for each of the nodes I have to have a receptor. That means for the ith node, I have to have receptor ti so i varies from 1, 2 up to M and the ith hidden layer node, I have to have sigma i again, i varying from 1, 2 up to capital M. So, I have to have Mm number of receptors, I have to have M number of variances or standard deviations.

So, this can be done in various ways, the simplest way is at a random from the training samples, you pick M number of samples, which are to be used as deceptors, which is not very logical rather, what appears to be more logical is, you go for clustering of this capital N number of training samples into M number of clusters. So, when I go for clustering, I basically get a space where it is more likely to have a feature vector. Clustering we have not discussed yet, but I will just for the purpose of completion of this lecture, I will simply indicate what does this clustering mean. So, I have capital N number of samples, from that I have to form capital M number of clusters.

So, it is understood over here, that this N has to be greater than capital M, if N is less than M that is number of samples training, samples is less than the number of clusters that are to be formed, it is not possible. Because, even if in a single cluster I assume that, there has been only one feature vector, at the most I can have N number of feature vectors or at the most I can have N of clusters.

So, I cannot form N number of clusters which is more than the number of feature vectors, which is provided. So, obviously our assumption is that the number of feature vectors N is greater than, the number of clusters that one has to form that is M. Now, there are various clustering algorithms, one of the clustering algorithm is say initially, I partitioned this N number of samples into capital M number of clusters. So, at every cluster will have a number of samples, for each of the created clusters say.

(Refer Slide Time: 47:39)



Suppose I have got a number of feature vectors, which are given like this. So, these are the set of feature vectors now, suppose I want to form three clusters so initially what I do is at random arbitrarily you divide this set of samples. You partition the set of samples into three different clusters so this cluster contains these samples, this cluster contains these samples, and this cluster contains these samples and so on. Now, once I have these initial clusters then what I can do is for each of the cluster so formed, I compute the cluster centre. So, which will be that representative of that cluster of the samples belonging to that cluster.

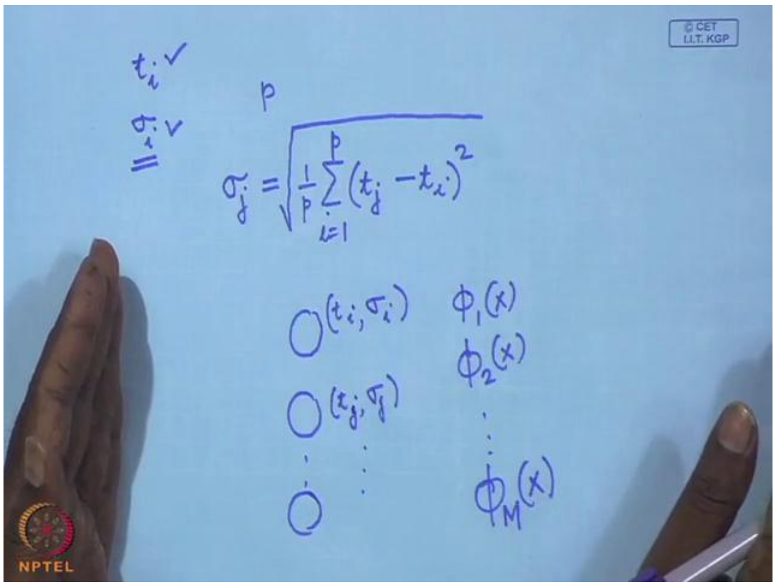
So, if over here suppose the centre of this cluster comes out to be somewhere over here, the centre of this cluster comes out to be somewhere over here and the centre of this cluster comes out to be somewhere over here. This will be the centre cluster, centre mu 1, this will be the cluster centre mu 2, and this will be cluster centre mu 3 and so on. Now, once I form this cluster centres in the next iteration what I do is for each of the samples, I find out which cluster is nearest to that particular sample. So, here you find that it may be possible that this class, this particular sample say this particular sample it's distance from mu 3 is less than its distance from mu 2.

So, initially though this sample was put to cluster 2, in the second iteration because its distance from the cluster centre 3 is found to be less than, its distance from the cluster centre 2, this sample will be more from cluster 2 to cluster 3. And this operation you will perform for every sample so for each such sample you find out that what its nearest cluster centre is and you move this sample to, the cluster whose centre is nearest to it. And you perform this for each of the samples so at the end of second iteration you will find that, initial clusters that we had formed, the samples in the initial cluster have changed. I still have three numbers of clusters, but the sample values have changed.

So, in the second iteration at the end of second iteration, again I compute that with this new set of samples what is its cluster centre. After computing the computation of this cluster centres, again I go for the re assignment of this feature vectors, based on its distance from this new cluster centres. And this operation you will perform in a number of iterations.

And finally, when it stabilises then we say that the clustering is complete and at that point whatever cluster centres I have, those cluster centres, have those cluster centres are my receptors. So, by this what I have done is, for each of the radial basis functions, or each of the nodes in the hidden layer I have computed the receptors, this receptors are nothing but the cluster centres.

(Refer Slide Time: 51:55)

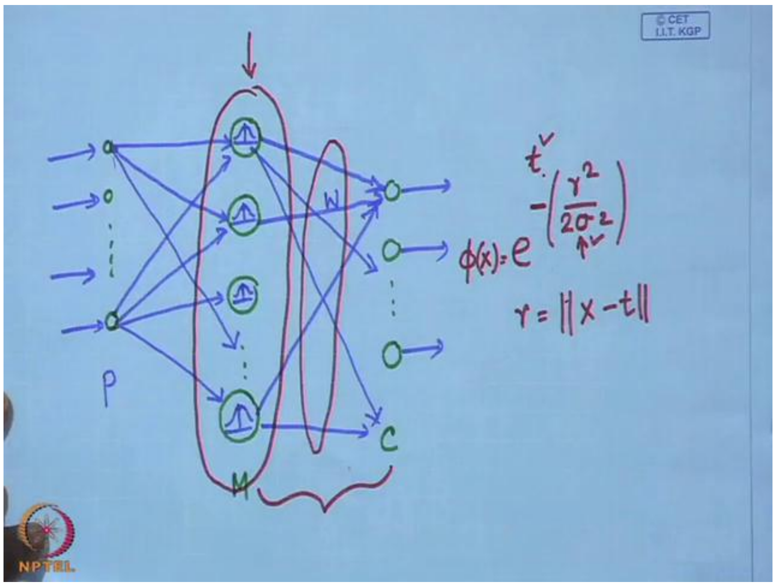


Once I do that, next what I have to do is... I have to, so for the every ith node I have to compute a ti, I found out what is ti. Next I have to find out what is sigma i so then what I can do is, I can go for a nearest neighbour tool. From every ti, I can find out P number of nearest receptors and the distance of those P numbers of nearest receptors. So, I get P number of distances and root mean square of all these distances, I can take as, that as the variance or the standard deviation sigma i. So, if I am interested to find out what is sigma j, what I will do is, I will take tj minus ti, take the square of this, sum of this for i is equal to 1 to P, as I have P number of nearest ti's 1 upon P of this and square root of.

This is one of the way in which I can compute sigma so I have the receptors ti and I have sigma i. So, once I have this ti and sigma i then you will find that in the middle layer every hidden layer node is defined, for every node I have ti, for every node I also have sigma i. So, here also have ti, I have say tj, here also have sigma j so this is defined for every node.

Now, if I feed any feature vector x to this, I have the corresponding phi q of x, I have the corresponding phi 2 of x, I have the corresponding phi M of x. So, the original feature vector x is now converted to M dimensional feature vectors now, using this feature vectors I have to go for classification. So, for that I have to train the output layer of this radial basis function neural network because as you see, if you remember the architecture of the radial basis neural network over here.

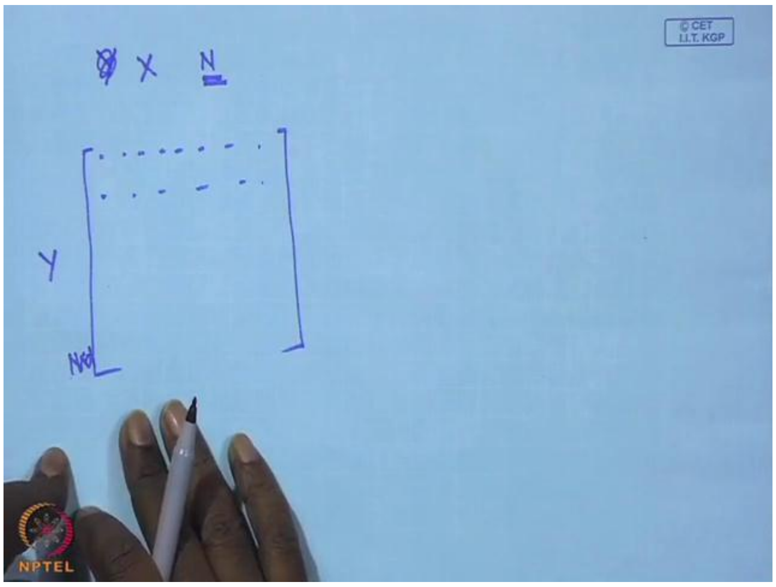
(Refer Slide Time: 54:37)



I had a set of weights W, so I have to compute this W for classification purpose. Now, here as a side that, as I increase the dimensionality and I, as I imposed a non-linear transformation and increase of dimension and increase of dimensionality taken together. That increases the possibility, that the classes will be linearly separable in the higher dimensional space. And as I increase the dimension, as I increase the value of M, the possibility of linear separability goes on increasing.

But I do not know that, how many such radial basis functions I have to use so that, my original problem becomes linearly separable. So, though I have increased the possibility of linear separabilty, I have not been able to guarantee that, they are linearly separability. So, the best way to find out this connection weights at the output layer is, by means of least mean square error technique. So, what we have is, we have already discussed about the LNS technique.

(Refer Slide Time: 56:14)



And therefore, we have seen that if Y is a set of feature vectors and if I have a set of feature N number of feature vectors Y then using this Y I can form a matrix. So, let me say this feature vector is X so if I have N number of feature vectors, every feature vector when I put in the form of a row, I have a matrix of say N by d, if d is the ((Refer Time: 56:43)) of the feature vectors.

And we call this matrix as Y and the logic that I put is, when I want to train say, fast output node of my classifier, only the samples if I freed sample which belongs to cross one, only that output should be 1, the rest of output should be 0. So, for every input feature vector I can define and output vector so using each of these outputs, I can define what is, what in case of this LMS algorithm, we have said as a bias vector. And then by using the pseudo-inverse technique, I can find out what is the output to it. So, this discussion I will stop here today, I will continue with this discussion in our next class.

Thank you.