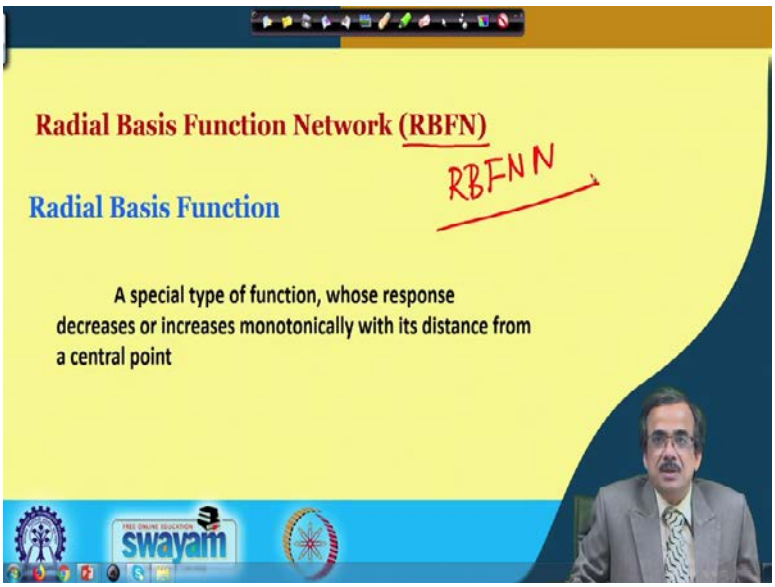
Fuzzy Logic and Neural Networks
Prof. Dilip Kumar Pratihar
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture – 25
Some Examples of Neural Networks (Contd.)

(Refer Slide Time: 00:15)



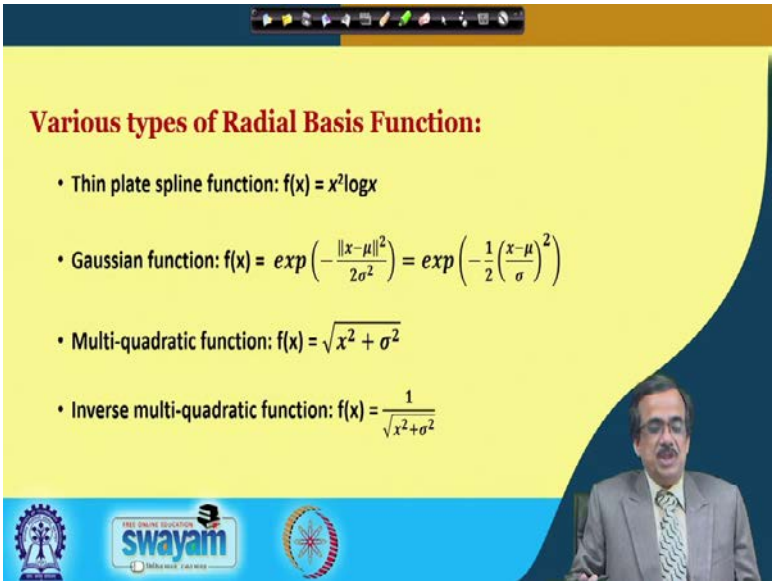
We are going to discuss the working principle of another very popular network and that is known as actually a Radial Basis Function Network, that is RBFN or Radial Basis Function Neural Network, that is your RBFNN. So, let us see how does it work, and how can it solve the input-output modeling problem.

That means, if you want to represent the input output relationship of a particular engineering system or a process, we can also use the radial basis function network in place of the multilayered feed forward network. Now, let me discuss first the working principle of this radial basis function network and then, I will make a comparison of this particular network with the multilayered feed forward network.

Now, to define like what do you mean by the radial basis function network, this radial basis function is actually a special type of function, where the response increases or decreases monotonically with its distance from a centre point. Now, I am just going to or take the example of a few radial basis functions and these radial basis functions are used

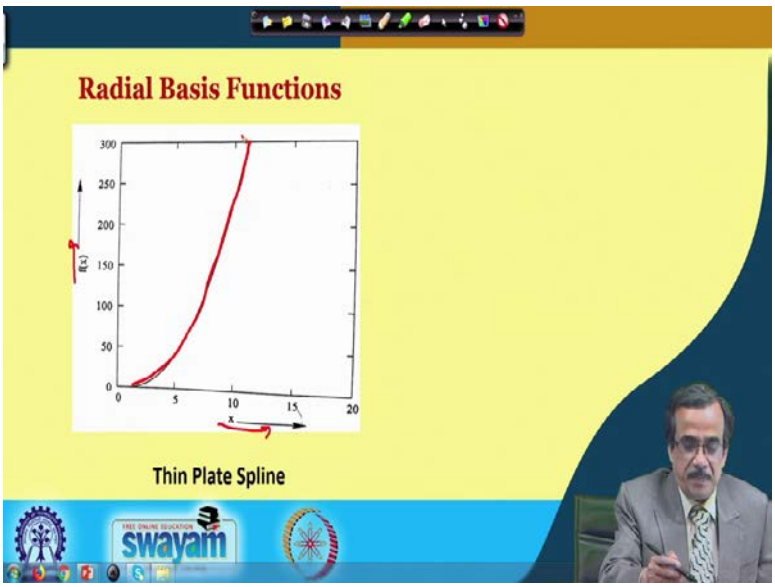
as the transfer functions in radial basis function neural network. Now, let me concentrate more on this particular the radial basis function.

(Refer Slide Time: 02:11)



Now, if you see the radial basis function, we have got the different types of function. For example, say we have got the thin plate spline function and mathematically, this is nothing but $f(x) = x^2 \log x$.

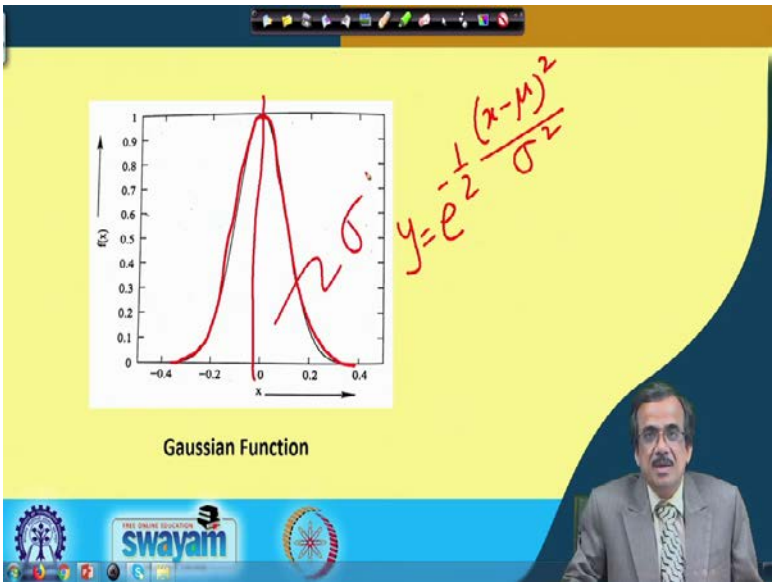
(Refer Slide Time: 02:33)



Now, if you see the plot of this particular function, the plot looks like this. So, this is your x, and this is f (x), that is y. And, here you can see, as x increases, so this particular function increases monotonically. So, this is actually a very good example of a radial basis function and this is known as the thin plate spline function.

Now, if you see the other forms of this particular radial basis function, we have got another function that is the well-known Gaussian function and this particular Gaussian function is used very frequently in radial basis function network. And, if you see this mathematical expression of the radial basis function, so this $y = f(x) = \exp(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2)$. Now, if you see this particular plot this figure shows the Gaussian distribution.

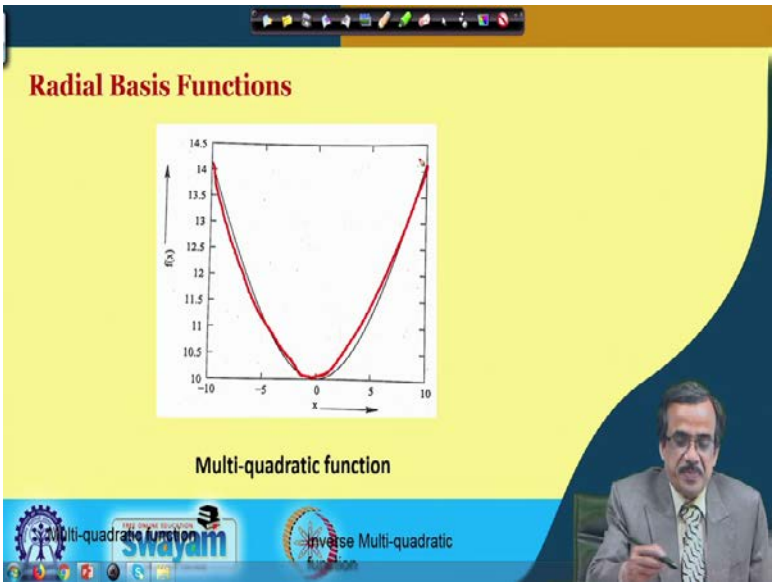
(Refer Slide Time: 03:43)



This figure shows a Gaussian distribution and this Gaussian distribution is used as a radial function and if you see the mathematical expression for this Gaussian distribution $y = \exp(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2)$. Now, if you see, so this is the way the function is actually increasing and this is a way it is decreasing. Now, if I select, this particular sigma, so I can actually control the distribution of this particular function. So, this indicates the mean property and sigma is going to indicate what should be the distribution.

Now, if I consider the smaller value for this particular σ, so I will be getting the steeper curve. On the other hand, if I consider the higher value for this particular σ, you will be getting the flatter curve in the radial basis function network. The next is your multi quadratic function, that is, $y = f(x) = \sqrt{x^2 + \sigma^2}$.

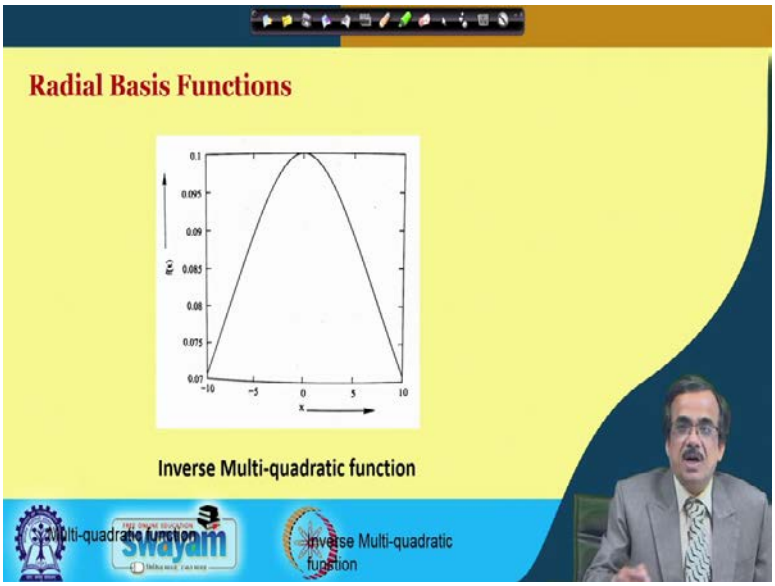
(Refer Slide Time: 05:47)



Now, if you see the plot of this particular function, this is nothing but the plot. Now, here, you can see that this particular function is decreasing monotonically and then, it is increasing also, ok. So, this is a very good example of a radial basis function.

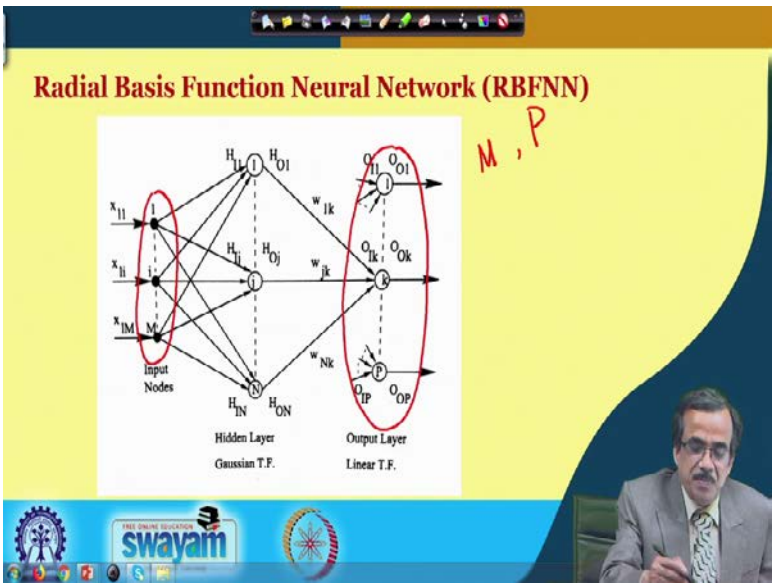
And, this is also very frequently used in radial basis function network. Now, then comes your the inverse of this, that is known as inverse multi quadratic function and $y = f(x) = 1/\sqrt{x^2 + \sigma^2}$. So, this is nothing but inverse multi-quadratic function. And, this is also very frequently used in your radial basis function network, ok.

(Refer Slide Time: 06:39)



And this is the function plot, the function plot for inverse multi quadratic function. Now, this radial basis functions are used in radial basis function network.

(Refer Slide Time: 06:55)



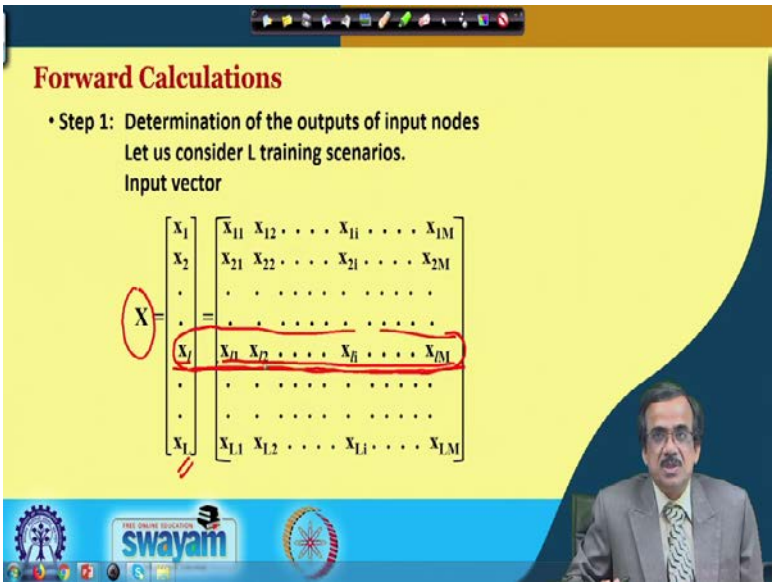
Now, if we concentrate on the structure of a radial basis function network, it looks like this. Now, supposing that I have got a process having say capital M number of inputs and I have got the capital P number of outputs.

Now, here on the input node actually we pass all the input parameters. So, these are all input parameters and on the output layer, we represent this output neurons, ok. And, in

between the input nodes and the output layer we have got a hidden layer, and the architecture of the topology of this particular radial basis function network depends on the number of neurons we put on the hidden layer and generally, for this radial basis function network, we use only one hidden layer. Now, here, how to decide the number of neurons to be put on the hidden layer that I am going to discuss in details.

But, before that let me tell you one fact regarding this particular network, then once again, I will be discussing how to decide the topology or the architecture of this particular network.

(Refer Slide Time: 08:25)



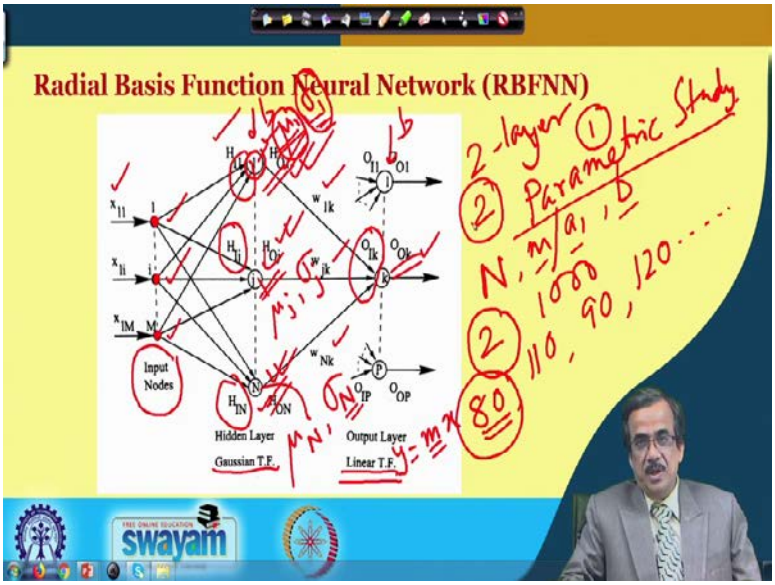
Now, let me concentrate on the training scenarios, first. Now, supposing that we have got say capital L number of training scenarios. So, this capital X is nothing but a collection of all capital L training scenarios. Now, if I concentrate on a particular training scenario, so l-th training scenario. So, capital X_small l is a collection of X_l1, X_l2, dot dot dot X_li, then a few other values at the last value is X_lM. So, this is actually nothing but the l-th training scenario; that means, the input or the l-th training scenario and supposing that I have got capital M number of inputs. So, this X l that is the l-th training scenario, so its inputs are nothing but these.

So, these inputs we are going to pass to the network and will be getting some calculated output. Now, this calculated output will be compared with the target outputs to determine the error values. Now, let us see, how does it work and how to decide, this particular

your architecture first and then, I will see how does it work how to carry out the forward calculation or how to carry out the feedback calculations or the feed forward calculations rather.

Now, as I told, the architecture depends on the number of the neurons you put on the hidden layers and at each of this particular hidden neurons, we put the radial basis function as actually your the transfer function. For example, say here, I am using some sort of the Gaussian distribution as the transfer function.

(Refer Slide Time: 10:37)



So, corresponding to the first hidden neuron, I have got say one Gaussian distribution and supposing that it is denoted by μ_1 and standard deviation is your σ_1. Similarly, for the j-th one, mean is μ_j and your standard deviation is σ_j. Corresponding to this n-th one, so the mean is your μ_N and standard deviation is nothing but is your σ_N.

Now, how to decide the value of this particular the capital N that, I am going to discuss. But, before that, let me tell you one more thing that here in place of input layer, I am putting input nodes. So, truly speaking for this particular network, there is no input layer and this is nothing but actually a 2-layer network. That means, here it has got one hidden layer, which is in between the input nodes and output layer and we have got one output layer. And, truly speaking there is no input layer here, instead what you have got it is your input node and if you notice it carefully to represent a neuron, I am using a circle

something like this and to represent the node, in fact, it is actually the filled-up small circle sort of thing, ok.

So, this indicates the node, this is node but not the neuron. So, this is a node not the neuron. And, the difference between this particular node and the neuron is, we do not use any transfer function here. Now, whatever is coming as inputs the same input you pass it here, so the input is passed through this particular node and all such inputs are summed up here. So, this is nothing but H_I1, that is the input of the first neuron lying on the hidden layer this is H_Ij that is the input of the j-th neuron lying on the hidden layer and this is H_In that is nothing but the input of the n-th neuron lying on the hidden layer.

Now, depending on this Gaussian transfer function, I will be getting some output here, some output here, some output here and to determine the input of the output layer by following the same principle. So, I will have to multiply this particular output by this connecting weight, this particular output by that connecting weight, and this particular output by this connecting weight and you sum them up, so you will be getting the input of the k-th neuron lying on the output layer and generally, on the output layer, we use the linear transfer function.

So, output is nothing but the input. So, output of the k-th neuron lying on the output layer is nothing but the input of the k-th neuron lying on the output layer. So, this way actually, it works. But, let me tell you one more thing and I have not yet discussed, in fact, how to decide the topology or how to determine the number of hidden neurons. Now, if you see the literature you will find that there are different ways, there are different methods used to decide what should be the number of hidden neurons in this hidden layer. Now, out of all such methods I am just going to discuss a few very popular methods.

For example, say this I have already mentioned the minimum number of neurons in the hidden layer has to be once again 2, but what should be the optimal number? To decide the optimal number, as I discussed, I can carry out some sort of parametric study, the way I carried it out for the multilayered feed forward network, and in the parametric study, what you can do is, you can decide what should be the optimal number of N, what should be your this particular coefficient of this transfer function.

For example, linear transfer function it could be, $y = mx$, what should be the suitable value for this particular m or if I use some non-linear transfer function like log sigmoid or say tan sigmoid, so in place of m, I will have to find out what is a_1 or a_2 and so on. And, of course, I can add some bias value, so I can put that bias, I can put some bias value, here, ok; so bias can also be determined. And, exactly the same procedure, which I discussed, we can follow just to find out what should be the near optimal values for this N, then comes m, a_1, a_2, b and all such things.

And, once you have got this near optimal network, now you can believe that particular the network. So, parametric study is one method, now I am just going to mention another very scientific method, which I have already discussed. So, what you can is, you can do some sort of clustering using the principle of fuzzy clustering. For example, say the fuzzy C-means clustering, then comes your fuzzy entropy-based clustering.

Now, if you do clustering based on similarity of the training data, there is a possibility of determining N, supposing that I have got 1000 training data, and if I do clustering based on similarity, what will happen is, supposing that I am getting ten optimal number of clusters for each cluster, supposing that say in the first cluster, say I have got say 80 data, second cluster I have got say 110 data, third cluster I have got say 90 data, fourth cluster I have say 120 data, and so on. Similarly, I have got say 10 clusters, 10 clusters mean your 10 hidden neurons.

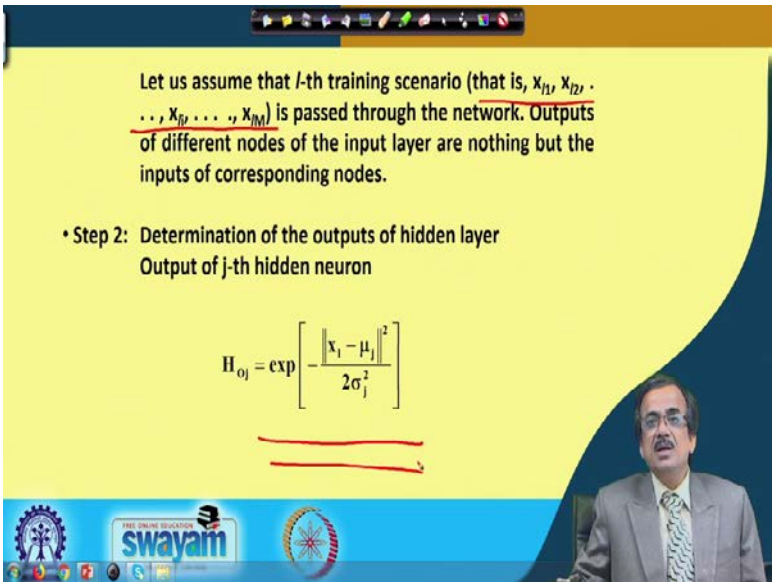
Now, each neuron is going to represent a particular cluster and say in the first clusters, I have got 80 data and based on its leader, the cluster centre, I can find out what should be the mean property of this particular Gaussian distribution. And, once I know this particular mean property, so I can find out the variance of this particular surrounding data and I can also find out, what should be the standard deviation, that is, σ_1. So, for each of these particular hidden neurons, the Gaussian distribution, I can find out, what should be the mean and standard deviation.

And, once you have got the mean and standard deviation for these 10 number of hidden neurons, I know all such properties and the moment I pass these particular inputs to these hidden neurons, depending on this your μ and σ, I will be getting the different outputs, although the inputs for each of these particular hidden neurons are exactly the same numerically.

For example, say what you are doing; this H_I1 is numerically exactly equal to H_Ij and that is numerically equal to H_IN. So, what I do is, we can find out this particular inputs and depending on the σ and μ, I will be getting the different output, different output, different outputs, and then, as I told, the outputs will be multiplied by the corresponding connecting weight and these are summed up here and then, it will pass through this particular transfer function just to find out the final output.

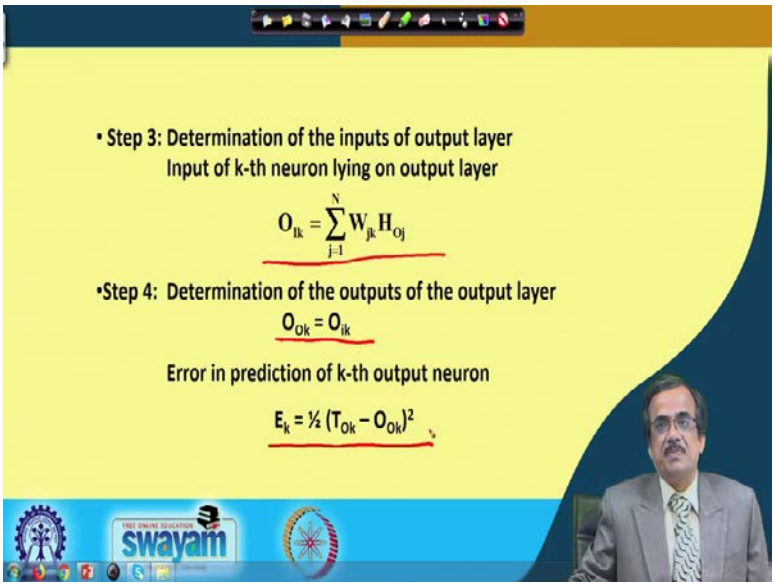
So, this is the way actually, this particular radial basis function network works. Now, whatever I have discussed, the same thing actually I have written it here.

(Refer Slide Time: 19:37)



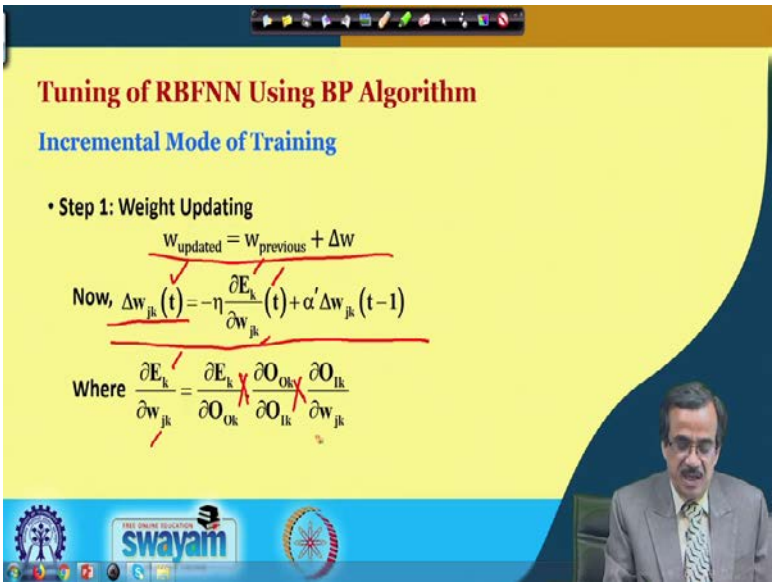
For example, say I am passing the training scenario, the l-th training scenario having your M numerical values for M inputs. And, once you got, I can find out the output of this hidden layer and the output of the hidden layer is nothing but this. So, H_Oj, that is, output of the j-th neuron lying on the hidden layer can be determined using this particular Gaussian distribution.

(Refer Slide Time: 20:13)



Now, if I have got output of this hidden neuron, what you can do is, very easily, I can find out the input of the k-th neuron lying on the output layer, and it is nothing but $O_{Ik} = \sum_{j=1}^{N} w_{jk} H_{Oj}$, and once I have got this, now, here we are using the linear transfer function. So, output is nothing but the input. So, I can find out the error in prediction at the k-th output neuron. So, that is nothing but this.

(Refer Slide Time: 20:51)

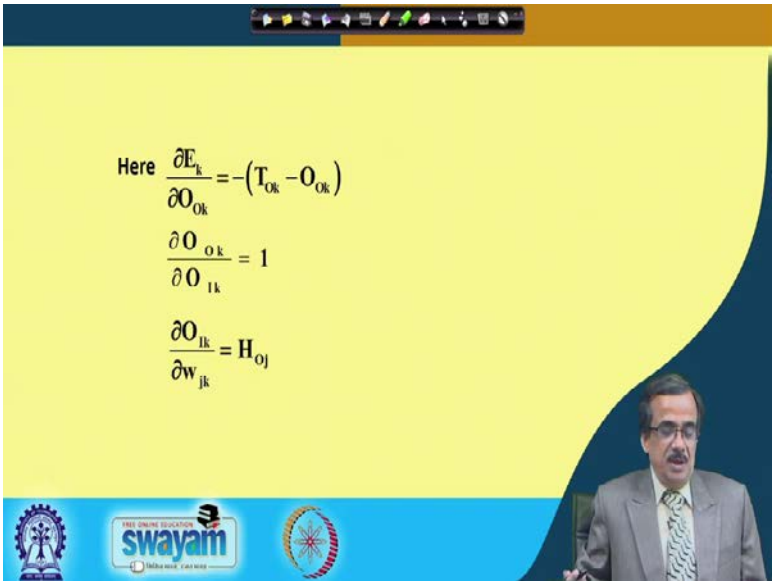


And, once you have got this particular output, what you can do is, you can use the incremental mode of training, just to update that particular network, and if you want to update this particular network very easily you can do the principal I have already discussed

So, here w_updated is nothing but w previous plus Δw, where this $\Delta w_{jk}(t) = -\eta \frac{\partial E_k}{\partial w_{jk}}(t) + \alpha' \Delta w_{jk}(t-1)$. So, I am using the generalized delta rule. And this particular partial derivative can be determined by following the same principle, the same chain rule of differentiation.

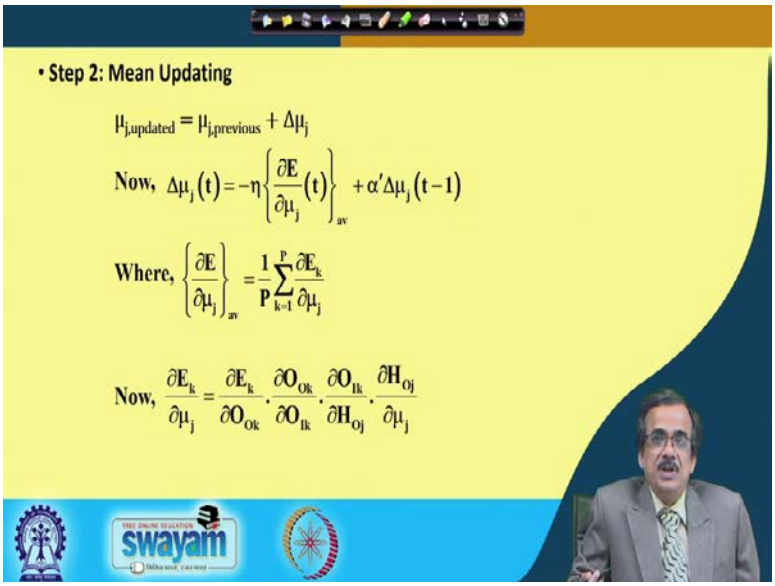
So, the $\frac{\partial E_k}{\partial w_{jk}} = \frac{\partial E_k}{\partial O_{Ok}} \frac{\partial O_{Ok}}{\partial O_{Ik}} \frac{\partial O_{Ik}}{\partial w_{jk}}$. So, by following this, I can find out, so this particular partial derivative.

(Refer Slide Time: 22:15)



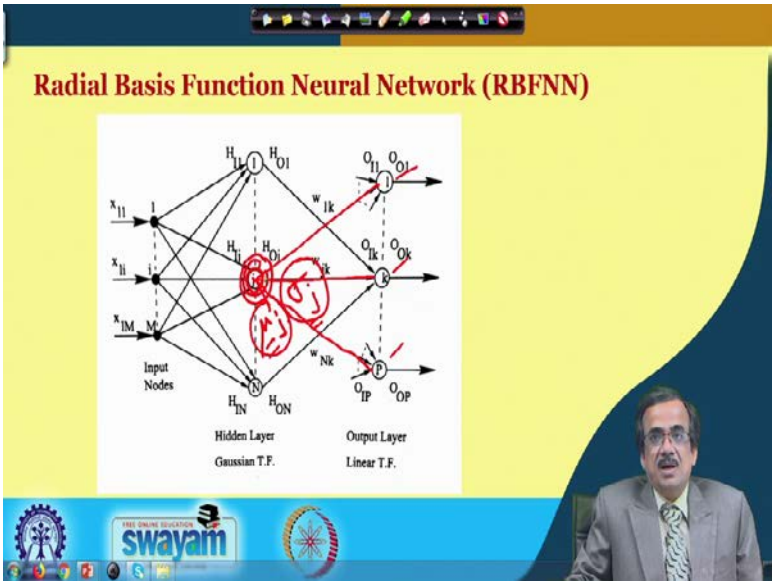
Now, the expression for each of the partial derivatives very easily you can find out, which I have discussed several times. So, we can find out the partial derivative of E_k with respect to O_Ok this is nothing but this particular expression, then partial derivative of O_Ok with respect to O_Ik is equals to 1, partial derivative of O_Ik with respect to w_jk is nothing but H_Oj.

(Refer Slide Time: 22:49)



So, this is the way you can find out. Now, let us see how to update the mean of this Gaussian distribution and how to update the standard deviation of this particular Gaussian distribution, that I am going to discuss. Now, before I go for this, let me once again concentrate on this particular network.

(Refer Slide Time: 23:11)

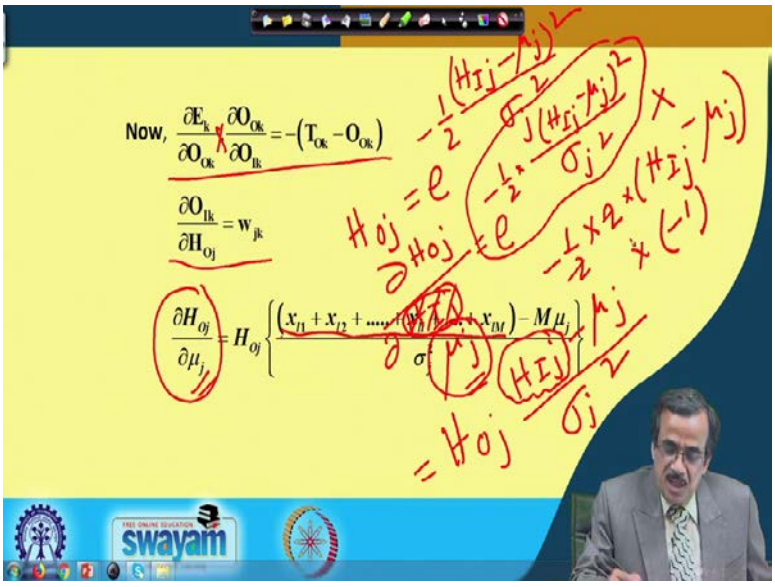


So, our aim is to find out, what should be the updated value for the mean. So, let me concentrate on this particular the radial basis function, the Gaussian radial basis function, and supposing that it has got the μ that is the mean and it is got the standard deviation,

that is your σ_j, so how to update this particular mean and your standard deviation that I am going to discuss. Now, if you see this particular H_Oj. So, this is connected to through this connecting weight to the first output neuron. Then, comes your this is connected to this particular network, so this is nothing but w_jp, and so on.

So, this particular radial basis function has got some contribution to each of these particular output neurons. That means, if I want to update this particular mean or the standard deviation, I will have to consider the average effect of this particular error. And, that is why, the way I discussed in fact, we are going to consider the average effect of this particular. Now, here your μ_j updated is nothing but μ_j previous plus $\Delta\mu_j$. Now, $\Delta\mu_j(t) = -\eta\{\frac{\partial E}{\partial \mu_j}(t)\}_{av} + \alpha'\Delta\mu_j(t-1)$. Now, this particular $\{\frac{\partial E}{\partial \mu_j}\}_{av} = \frac{1}{P}\sum_{k=1}^{P}\frac{\partial E_k}{\partial \mu_j}$. Now, you are multiplied by k sorry summation k equals to 1 to P partial derivative of E_k with respect μ_j and this is multiplied by 1 by P. Now, this particular partial derivative can be determined using the chain rule of differentiation. So, by following the same procedure I can find out.

(Refer Slide Time: 25:49)



So, very easily you can find out. So, this particular expression, that is $\frac{\partial E_k}{\partial \mu_j} = \frac{\partial E_k}{\partial O_{Ok}}\frac{\partial O_{Ok}}{\partial O_{Ik}}\frac{\partial O_{Ik}}{\partial H_{Oj}}\frac{\partial H_{Oj}}{\partial \mu_j}$. Now, we will have to find out the partial derivative of

H_Oj with respect to your, this particular the mu_j. Now, let us try to understand how to get this particular the expression.

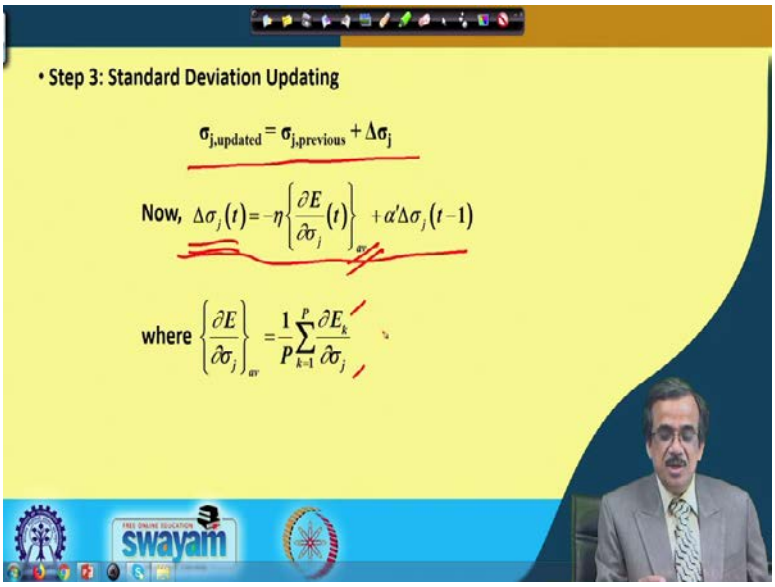
Now, this is a Gaussian distribution. So, if you write on the expression, that is, your H_Oj, this is nothing but e raise to the power minus half. Then comes your x minus μ, here x is what that is input that is H_Ij that is nothing but the variable x minus μ_j square divided by is your σ_j square. So, this is the Gaussian distribution.

Now, if I find out its derivative for example, say if I try to find out H_Oj with respect to your H_Ij. So, how to find out? It is very simple. So, what you can do is, this will become e raise to the power minus half multiplied by σ_j square, then comes your this H_Ij minus μ_j square multiplied by your minus half. Then comes your, so this is I am sorry this is actually your with respect to μ_j I am sorry for this. So, $\frac{\partial H_{Oj}}{\partial \mu_j} = e^{-\frac{1}{2}\times\frac{(H_{Ij}-\mu_j)^2}{\sigma_j^2}} \times(-\frac{1}{2})\times 2\times(H_{Ij}-\mu_j)\times(-1)$. So, this is nothing but is your the derivative with respect to your the μ_j.

So, let me repeat. So, partial derivative of H_Oj with respect to mu j is nothing but e raise to the power minus half, H_Ij minus μ_j square divided by σ_j square multiplied by minus half multiplied by 2, H_Ij minus μ_j multiplied by minus 1. Now, this can be written as what is this? So, this is nothing but is your H_Oj. So, this is your H_Oj, ok. Now, this H_Oj. Now, this 2, 2 gets cancel minus and minus this will become plus, so this is nothing but is your $\frac{\partial H_{Oj}}{\partial \mu_j} = H_{Oj}\frac{H_{Ij}-\mu_j}{\sigma_j^2}$. Now, this H_Ij is what? H_Ij is nothing but the summation of your x_l1, l2 up to your lm that is nothing nut is your H_Ij. If you remember the input of the j-th neuron laying on the hidden layer is nothing but summation of all such values. And, here, I am putting minus $M\mu_j$. Now, for each of the dimension, so I consider that it has got your mean μ_j and we assume that all the dimensions has got the same the mean value, which is an assumption and I have got M dimensions. So, this is nothing but minus $M\mu_j$ divided by σ_j square.

So, this is the way actually, we can find out this derivative, that means, you can find out actually, this expression for the partial derivative of H_Oj with respect to mu_j, so this particular the expression, we can find out very easily.

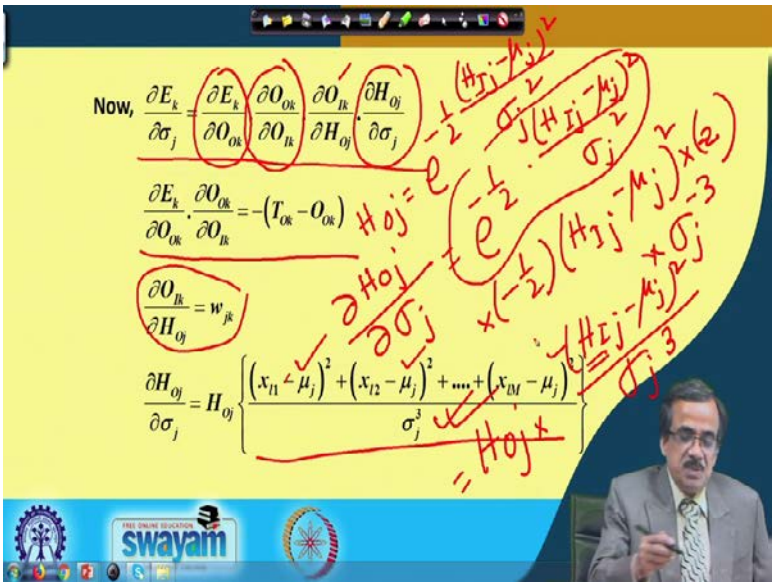
(Refer Slide Time: 30:59)



And, once you have got this particular thing, now, we are in a position to find out the change in μ_j and once you have got the change in μ_j, we can find out the updated value.

Now, the same principle, I am just going to use for updating your the σ. So, σ_j updated is nothing but is your σ_j previous plus $\Delta\sigma_j$, exactly in the same way I am writing down the expression of $\Delta\sigma_j$. So, will have to consider the average effect once again and the partial derivative of E with respect to σ_j average is nothing but 1 by P multiplied by summation k equals to 1 to P partial derivative of E_k with respect to your σ_j. Now, once again I will have to derive that particular the expression.

(Refer Slide Time: 31:53)



Now, this partial derivative of E_k with respect to your σ_j is nothing but this particular expression according to the chain rule of differentiation. And now, what you can do is, so this particular partial derivative of E_k with respect to O_Ok multiplied by partial derivative of O_Ok with respect to O_Ik this I can find out. Now, partial derivative of O_Ik with respect to H_Oj, I can find out.

Now, I am in a position to find out what should be this partial derivative, that is partial derivative of H_Oj with respect to your σ_j and this is nothing but is your this particular big expression. Now, very easily once again we can derive this particular expression. So, H_Oj is nothing but e raise to the power minus half then comes your x is nothing but say H_Ij minus your μ_j square divided by σ_j square this is a Gaussian distribution.

Now, if I find out the $\frac{\partial H_{Oj}}{\partial \sigma_j} = e^{-\frac{1}{2}\frac{(H_{Ij}-\mu_j)^2}{\sigma_j^2}} \times (-\frac{1}{2})(H_{Ij}-\mu_j)^2 \times (-2) \times \sigma_j^{-3}$, ok.

Now, if you simplify, this is nothing but is your H_Oj multiplied by actually this minus, minus becomes plus 2 and half. So, I will be getting like H_Ij minus μ_j square divided by σ_j cube, exactly the same expression I have written it here. Now, this H_Ij is what? H_Ij is nothing but x l1, l2, up to lm and this μ_j. So, for each of the dimension I am considering the same mean value mu_j. So, this is nothing but is your, x_l1, minus μ_j

square plus x_l2 minus μ_j square and the last term will be x_lm minus μ_j square divided by your this sigma_j cube. So, this is the way actually we can find out.

So, this particular partial derivative and once you have got this particular partial derivative, now, we are in a position to find out what should be your this particular the $\Delta\sigma_j(t)$ and once you have got this $\Delta\sigma_j(t)$, so I can update this particular your σ_j.

So, this is the way, actually the connecting weight, the mean and standard deviation of the Gaussian distribution used in the radial basis function for this network can be updated. And, through a large number of iterations, this particular network is going to give more and more accurate prediction, that is the better prediction, and this is the way, actually this radial basis function network is working.

Thank you.