Intelligent Systems and Control

Prof. Laxmidhar Behera

Department of Electrical Engineering

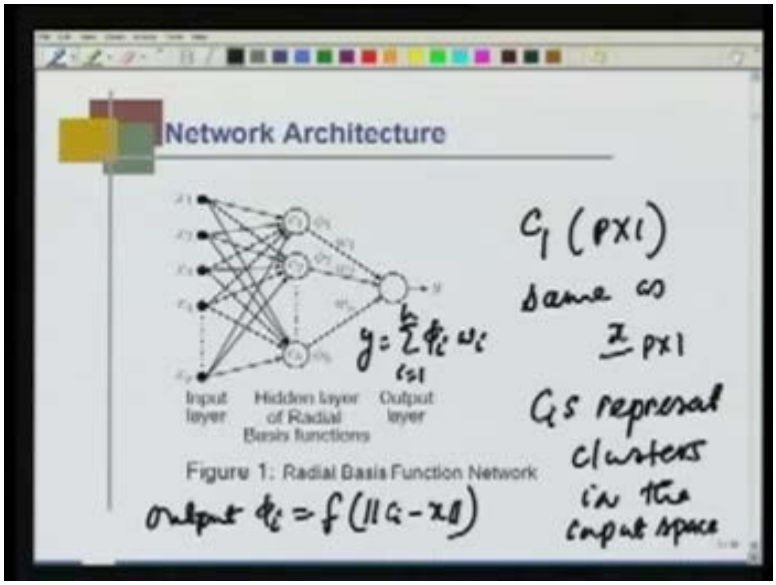
Indian Institute of Technology, Kanpur

Module – 1 Lecture – 6

Radial Basis Function Networks

This is lecture 6 in the first module of neural networks, a course on intelligent control. Today, we will take up another architecture of learning system that is used widely in control system. This is called radial basis function network.

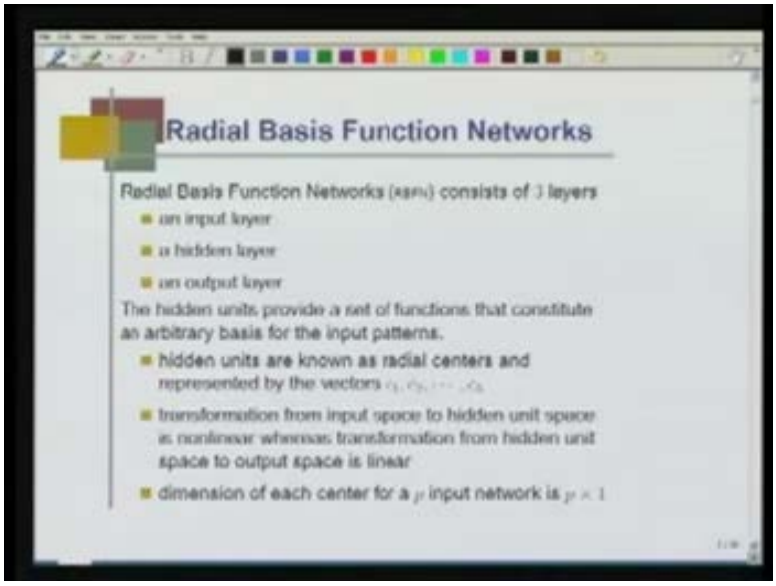
(Refer Slide Time: 00:56)



The network architecture is given here. You can see that it has three layers. The first layer is as usual the input layer. The second layer, which is the hidden layer, is different from the computational unit in the hidden layer – different in structure compared to the multilayer network that we discussed. These computational units are known as radial centers. It is a radial center because if I look at C_1, this is a p into 1 vector – the same as the x vector, which is p into 1, the input vector. These centers represent the clusters in the input space; C_1, C_2 and C_h represent the clusters in the input space. So, C_is represent clusters in the input space. The output of each center, which is phi_i, is a function of the Euclidean distance between C_i and x.

You can see that the computational unit has a different function than the multilayer network. Then what you are seeing is that we compute this phi_i. The C_is are known as radial centers; they represent small clusters in the input space. Then y, the output, is simply summation of phi_i w_i, i equal to 1 to h. This is the structure of a radial basis function network. There are certain advantages we will talk about. Let me mention what we talked about just now.

(Refer Slide Time: 04:00)

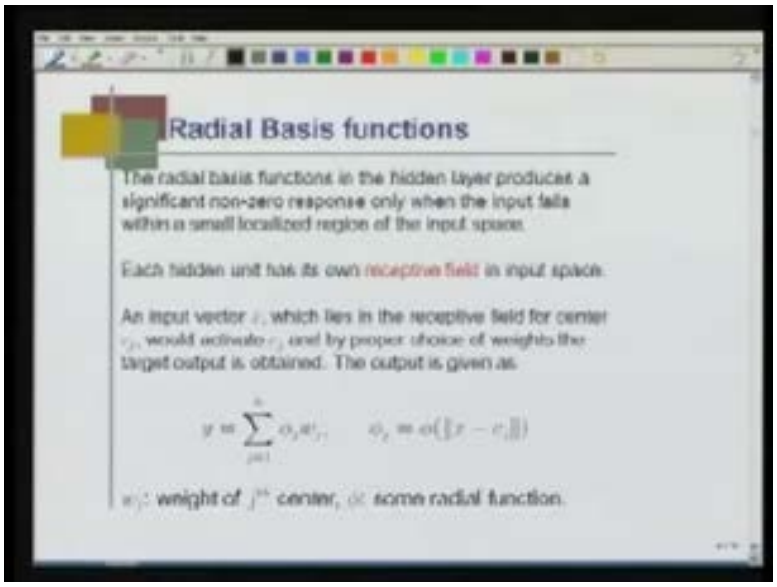


The radial basis function network consists of three layers: input layer, hidden layer and output layer. Unlike in multilayer network, you can have as many hidden layers, but here, you can have only one hidden layer. The computational units in the hidden layer have a different function than that of multilayer network. The hidden units provide a set of functions that constitute an arbitrary basis of input patterns. The hidden units are known as radial centers and are represented by the vectors C_1, C_2, … C_h and they have the same vector dimension as that of the input x.

Transformation from input space to hidden unit space is nonlinear, whereas the transformation from the hidden unit space to output space is linear – that is what we talked about (Refer Slide Time: 04:55). Here, the phi_is is the nonlinear transformation of x input. Actually, this is a function of the known or the Euclidean distance between C_i and x, which we normally represent by the second (Refer Slide Time: 05:14). This is nonlinear, phi_i is a nonlinear function of x_i, whereas the output y is a linear function of phi. That is the difference. The dimension of each

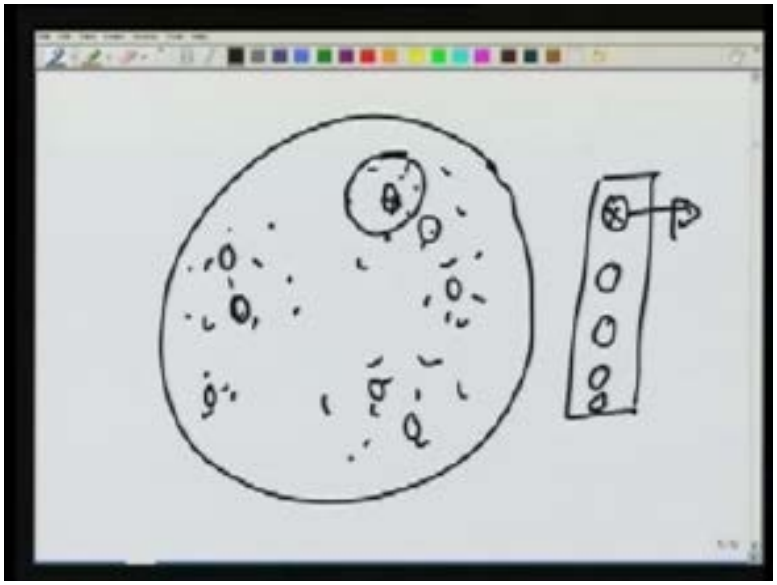
center for a p input network is p by 1 – I have already told you that. This is our network that we discussed.

(Refer Slide Time: 05:42)



The radial basis functions in the hidden layer produce a significant non-zero response only when the input falls within a small localized region of the input space. Let me explain this.

(Refer Slide Time: 06:00)



We are talking about input space. These data belong to this input space. The input space instead of being represented by each and every data, the principle in radial basis function network is say for example, the inputs, the data are all distributed in a (Refer Slide Time: 06:18) set like this. If this is the data, each point represents a datum in this input space, we can easily say let us select from this bulk of data two clusters here, another two clusters here, another two clusters here. These are small data points – one cluster, another cluster.

What is the meaning of cluster? This is another point that represents the data around this. Similarly, this is another point that represents the data around this. When the input data is very near this cluster, then the output of this cluster or output of this center will be maximum and the response of this radial center…. Let me first clarify what I am trying to tell you. What I am telling you is that we have a radial center and these radial centers are placed in the hidden unit. These radial centers are the clusters in the input space. Input space means a space where all possible data in the input are located.

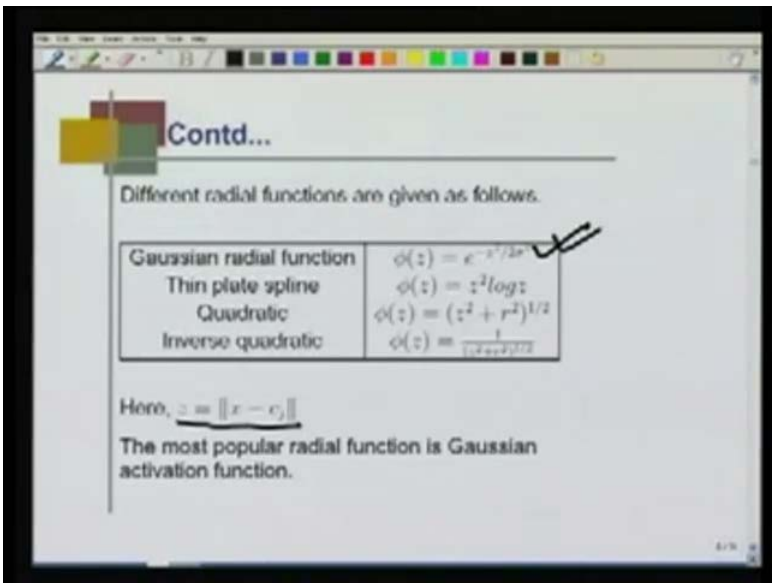
That means we are trying to construct a map from the input space to output space. In the input space, all possible varieties of data are there. The radial centers represent this data. In an ideal case, the input space may contain many data – not infinite, but many data. To represent this data using finite data points and these finite data points are known as clusters. Each cluster has its own receptive field in the sense that each cluster represents a certain data point and for that data point, that radial center output will be maximum.

I place these clusters in this hidden layer. For example, if I say this cluster is this cluster (Refer Slide Time: 09:24), then the output of this cluster or radial center will be maximum for the data lying in its receptive field. Obviously, the output of this for any other data in this away from this radial center will be minimum. That is the idea. The radial basis function in a hidden layer produces a significant non-zero response only when the input falls within a small localized region of the input space.

Each hidden unit has its own receptive field in the input space, which I just told you – the output will become maximum for the radial center when the input data belongs to its receptive field. An input vector x_i that lies in the receptive field for center c_j would activate at c_j and by proper choice of weights, the target output is obtained. The output is given as y equal to sigma phi_j w_j,

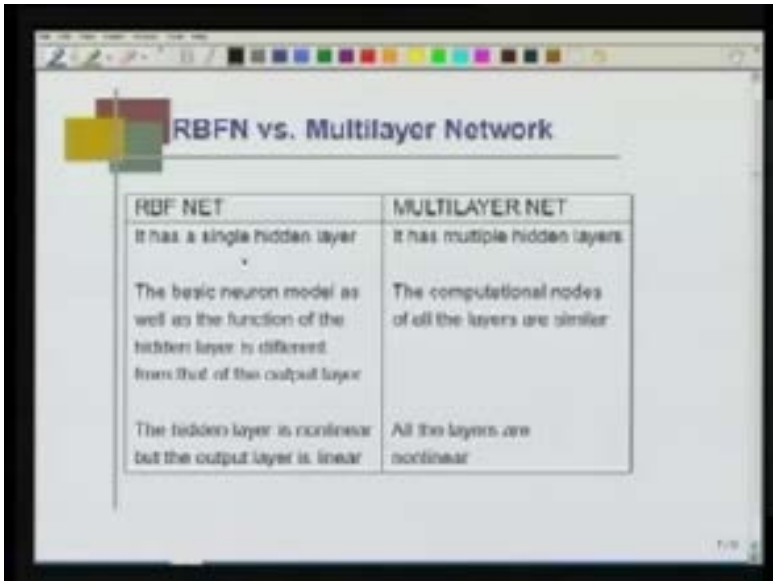
where phi_j is phi of x minus c_j known; actually, this phi represents a function – what I said earlier. As I have already told, the output y is a linear function of the outputs of the radial center, which is phi_j and phi_j is a nonlinear function of x, a nonlinear function of distance between x and c_j; phi is some radial centers.

(Refer Slide Time: 11:16)



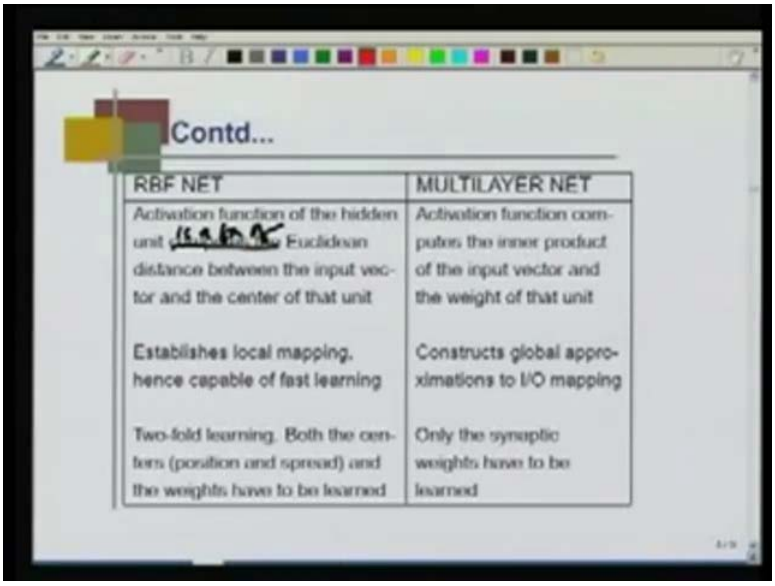
What are these radial functions? There are certain popular radial functions. If I define z as a Euclidean known between x, the input data, and the j th center…. We saw that we have h centers and the distance is Euclidean distance between the j th center and x. You can easily say that this is the Euclidean distance between x and c_j (Refer Slide Time: 11:46). The types of radial functions that we normally use are Gaussian radial function and thin plate spline function. Gaussian is very popular: e to the power minus z square by 2 sigma square, where z is the Euclidean distance between the data and the radial center. Similarly, the thin plate spline function is z square log z. phi z is z square plus r square to the power of 1 by 2 is a quadratic function and is the third one. The inverse quadratic is just 1 upon z square plus r square to the power 1 by 2, that is, root square of z square plus r square. The Gaussian function has been very popular while selecting the radial center.

(Refer Slide Time: 13:06)



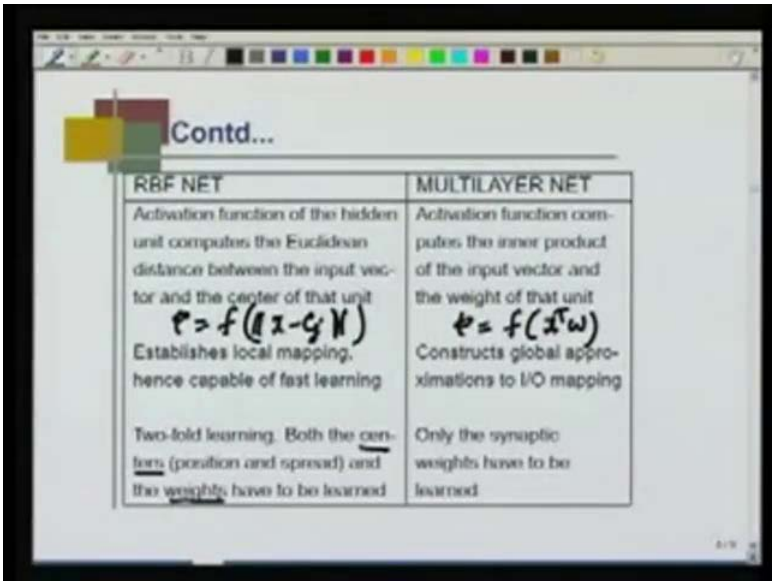
What is the normal difference between a radial basis function network and multilayer network? The difference is that a radial basis function network has a single hidden layer and a multiple layer network has multiple hidden layers. MLN has multilayer network and RBFN has a single hidden layer. The basic neural model as well as the function of the hidden layer is different from that of the output layer. In the output layer, we have simply summation, where the hidden computation is a function of the Euclidean distance between input and the center, whereas the computational units in multilayer networks are all similar. The hidden layer is nonlinear, but the output layer is linear. Here, all layers are nonlinear but not necessarily – sometimes in multilayer network also, we can make the output layer to be linear.

(Refer Slide Time: 13:53)



In the radial basis function network, the activation function of the hidden unit is a function of (Refer Slide Time: 14:15) of the Euclidean distance between the input vector and the center of that unit. The activation function is a function of the Euclidean distance between the input vector and the center of that unit in a radial basis function network, whereas in a multilayer network, the activation function computes the inner product of the input vector and the weight of that unit. In a sense, this is all right.

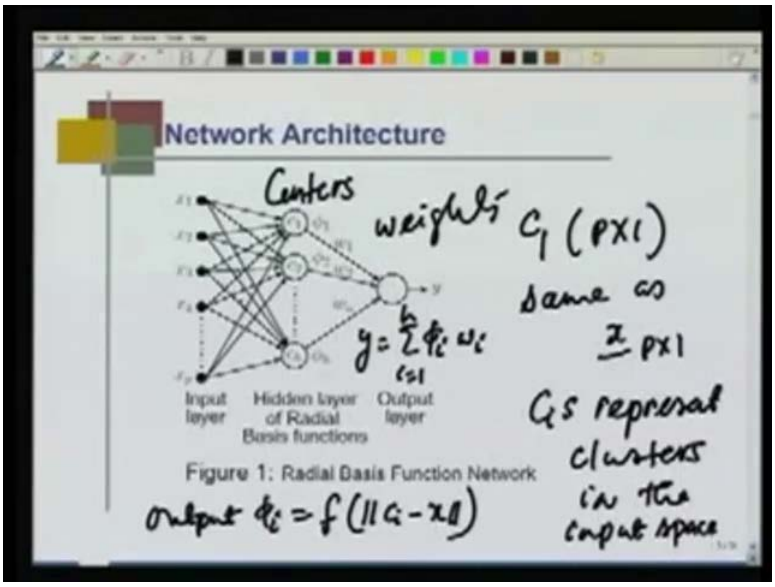
(Refer Slide Time: 14:45)



The objective here, whatever is written is just a comparison. What is happening is that the activation function is a function of the Euclidean distance between the input vector and the center, whereas this is the inner product – the activation function is a function of inner product. It is not activation function. The idea should be very clear to all of you that here, the function phi is a function of x minus $C_{j,}$ whereas this is the sigmoidal function here. If I say it is also phi, then in this case, it is f of x transpose w, the connection weight associated with the input.

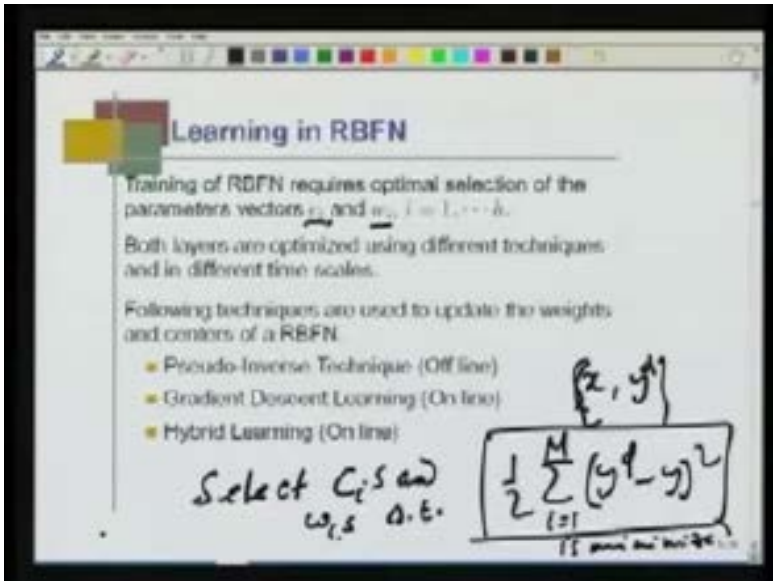
Such a weight is absent in this case, the radial basis function network. The radial basis function network establishes a local mapping and is hence capable of fast learning. The multilayer network constructs a global approximation to the input/output mapping. The learning in a radial basis network function consists of two different categories of parameters: one is the radial centers and the other is the connection weights in the output layer, whereas in the multilayer network, the only parameters are the synaptic weights.

(Refer Slide Time: 16:40)



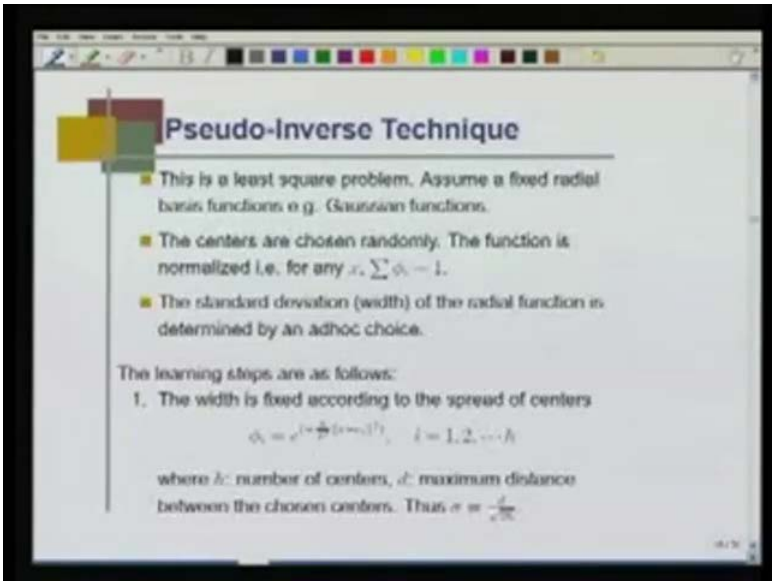
In a radial basis function network, the parameters are C_1, C_2 up to C_h – these are all centers (Refer Slide Time: 16:49) and w_1, w_2, up to w_n are weights. The weights are in the output layer and the centers are in the hidden layer.

(Refer Slide Time: 17:03)



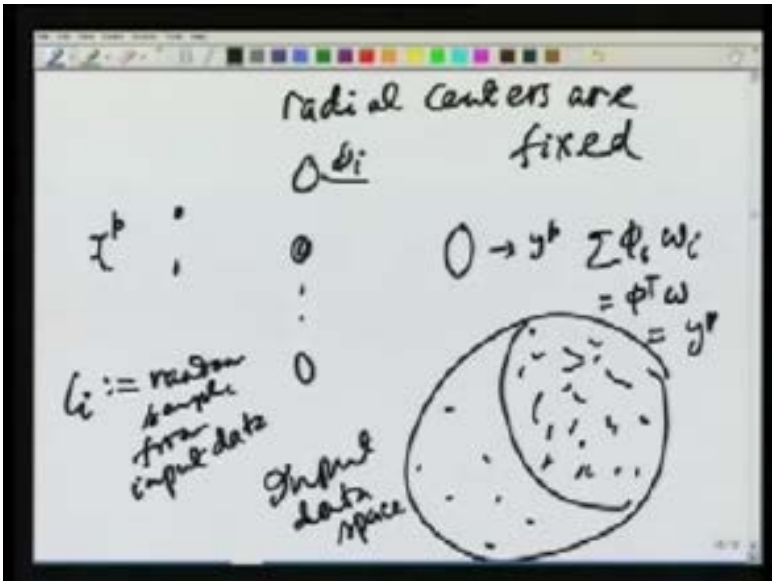
The training of RBFN requires optimal selection of the parameter vectors c and w, the weight vector. We have to select in such a way that given this set of data x and y d, the cost function half sigma over all patterns, i equal to 1 to M number of patterns into y_d minus y square (y_d is the desired output and y is the radial basis function output) must be minimized. The objective is select C_is and w_is such that this particular function is minimized. This is the objective. The following techniques are used to update the weights and centers of an RBFN. There are many methods by which these centers and weights are all optimized. One is the very simple pseudo-inverse technique, which is an offline technique; the gradient descent learning and hybrid learning are online techniques. Gradient descent is the same as back propagation that we talked about in multilayer network.

(Refer Slide Time: 18:53)



What is a pseudo-inverse technique? This is a least square solution. Let me tell you what we will do here.

(Refer Slide Time: 19:08)



We have inputs, then we have hidden units, then we have an output unit. Given your x p, we compute what is y p. What we do here in pseudo-inverse technique is that the radial centers are fixed. How do you fix the radial center? The radial center represents the clusters in input data space. This is my input data space. I have a stack of input data. What I can do is that I can

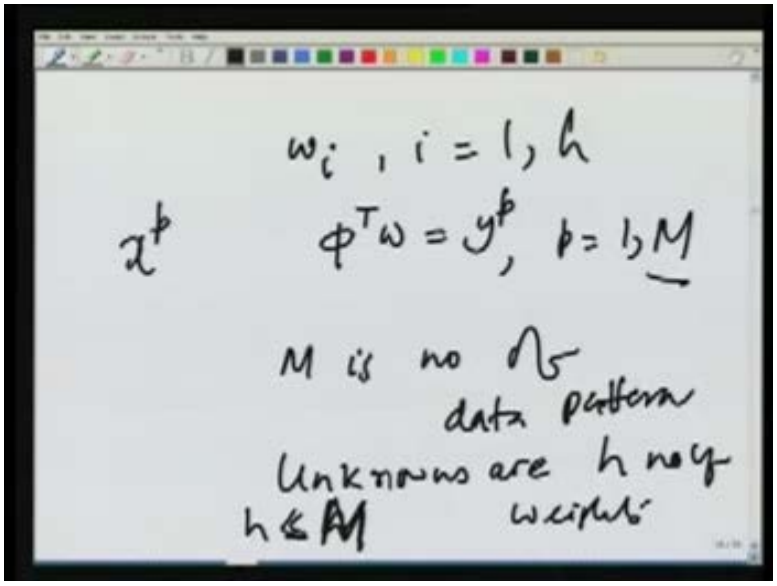
randomly sample this data and I assign each random sample from this data to C_is. What I do is I assign a random sample from the input space to C_i. From the input space or input data, I select a random sample and assign to C_i and like that, I assign all the centers.

The objective is that centers should be selected so that they are almost uniformly distributed in the data. That means wherever there are more data points, more data should come from there and less data should come from where there are less data points. For example in this zone, there are more data points (Refer Slide Time: 20:54) and so, more number of points should come and this should be assigned as a radial center. These data points are very sparsely located in this zone. Then, very few points should come and we assign as the radial center – that is the objective. That way, we can fix the radial center.

Once the radial center is fixed, then the outputs are known – these phi_is are fixed, because phi_is are based on whatever the center; centers are fixed, so phi_is are same, phi_i is fixed given a specific data point. Given a specific data point, what do we have now? I have to write like this.

phi is a vector, w also is a vector, so I have to write phi_i w_i. This is right, phi_i w_i is the same as phi transpose w. This phi_i w_i is now y p. For each x p, I can collect this kind of equation. So how many unknowns do I have? The ws are…. We can see here (Refer Slide Time: 22:31) the number of parameters now to be estimated are…. This is actually not n, this is h (Refer Slide Time: 22:39), w_h. The number of weights is h because the centers are h; the number of weights is h for a single output radial basis function network.

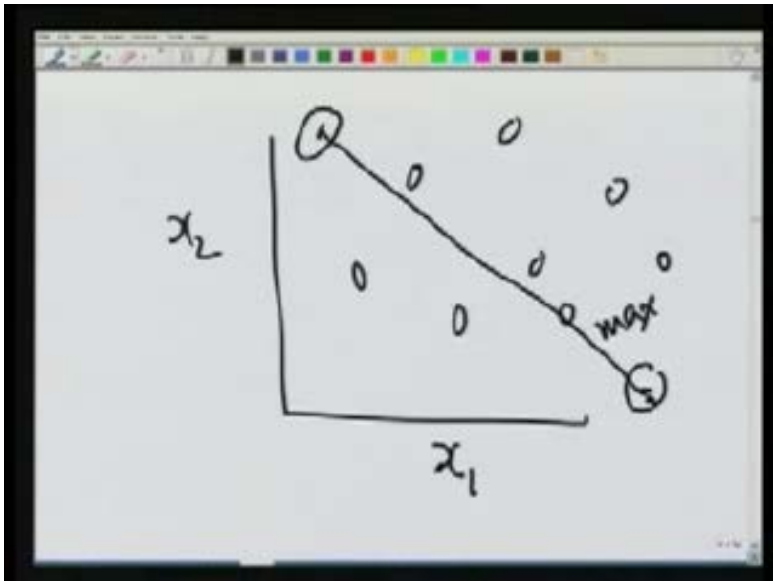
(Refer Slide Time: 23:02)



We have w_is, i equal to 1 to h. For a given input pattern, we find out this equation phi transpose w is y p. Like that, how many equations do we have? p is equal to 1 to M, where M is the number of data patterns. We have these many equations. How many unknowns? The unknowns are h weights – these are unknowns. Normally, the data patterns are many and h is usually much less than M. In essence, you have more number of equations but less unknowns. So, we can easily find a least square fit. That is the objective here.

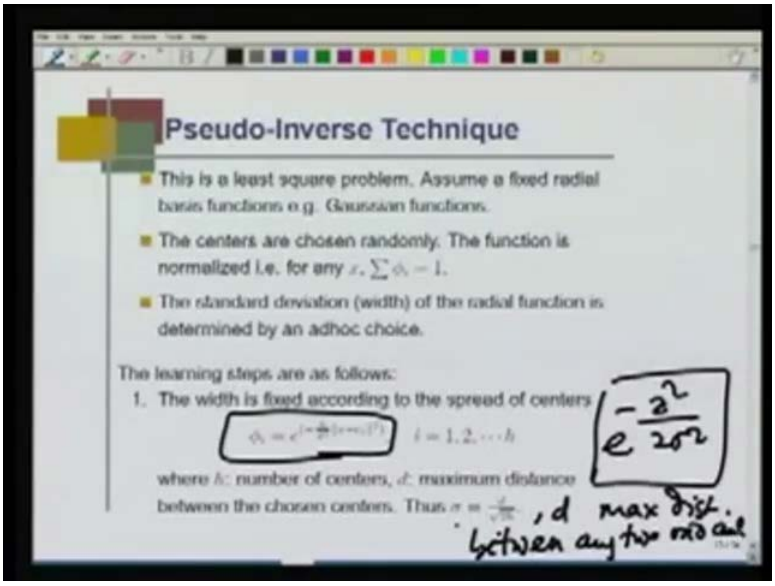
(Refer Slide Time: 24:28) This is a least square problem. Assume a fixed radial basis function, a Gaussian function. We assume that all the radial centers in the hidden units are Gaussian functions. The centers are chosen randomly as I said. The standard deviation of the radial function is determined by an ad hoc choice. The learning steps are as follows: the width is fixed according to the spread of the centers. Your phi_i is an exponential function (like a Gaussian function) e to the power of minus h upon d square and this is the known (Refer Slide Time: 25:13) – x minus c, the Euclidean distance between x and c_i square. This is the distance square. h is the number of centers and d is the maximum distance between the chosen centers. What is the meaning of this maximum distance between the chosen centers? If we go back (Refer Slide Time: 25:36)….

(Refer Slide Time: 25:40)



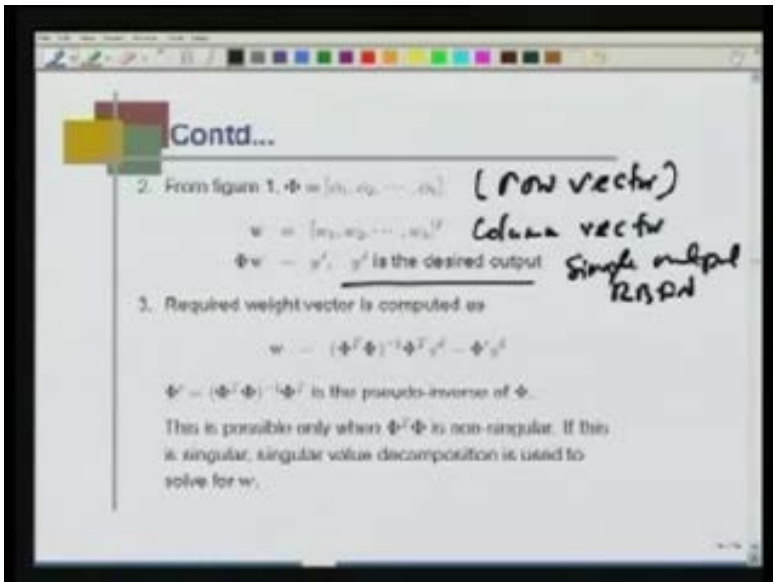
Let us say that we have two-dimensional data x_1 and x_2. I have a data point, one cluster here, another cluster here and all other clusters are like this. Obviously, for this cluster and this cluster (Refer Slide Time: 26:04), the distance between these is maximum. Similarly, once you have fixed the radial centers, find out two centers that have maximum distance between them.

(Refer Slide Time: 26:25)



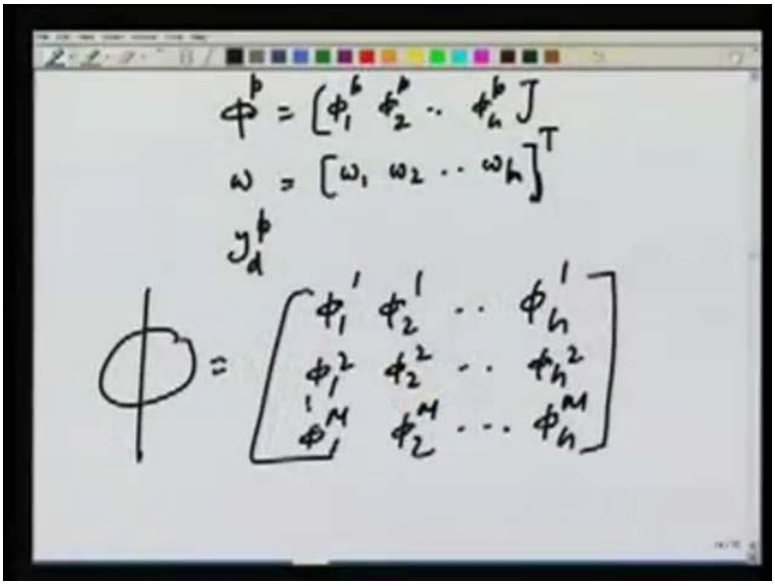
Once you do that, then that is d, that maximum distance is d. Obviously, if you look at that, if I write e to the power of minus x minus C_i, the Euclidean distance square by sigma square, then sigma will become d upon root 2 sigma squared. Normally, the function is like this: e to the power of minus z square by 2 sigma square. This is the normal form of a Gaussian function and if I represent this function as this (Refer Slide Time: 27:03), which is written here, Then obviously sigma is d upon root of 2 h and d is the maximum distance between any two radial centers.

(Refer Slide Time: 27:27)



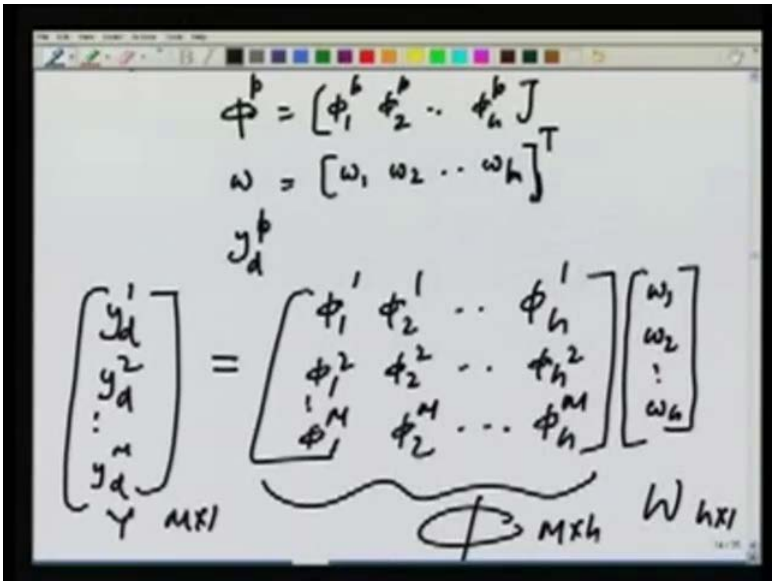
Now, we will formulate the problem. The problem is let us say phi is phi_1, phi_2 up to phi_h is a row vector that is taken here. This is a row vector phi. These are the outputs of the radial centers. w is the weight vector. This is a column vector. phi w is y d, where y d is the desired output. Now, we are only talking about a single output RBFN. You can also talk about multilayer output – it is the same, the formulation will be the same. The required weight vector is computed as…. What I am trying to do here is that we will find a pseudo-inverse technique, how to solve a radial basis function network.

(Refer Slide Time: 28:37)



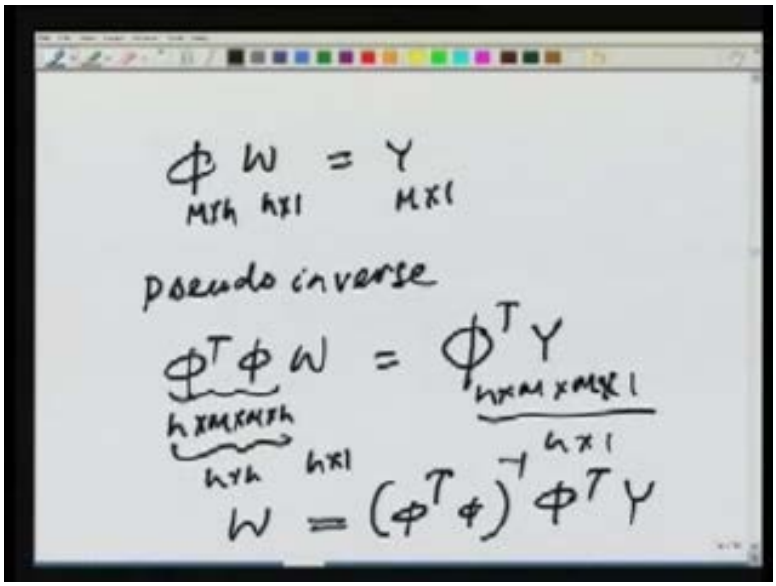
For a p th pattern, phi p is a vector of phi_1 p, phi_2 p up to phi_h p. These are the outputs of the radial centers. The weight is w_1, w_2 … w_h and the corresponding output is y_d p. Similarly, we have M patterns. If I write this in terms of matrix, I can write a capital phi, which is phi_1 1, phi_2 1 up to phi_h 1. These are the radial center outputs for the first pattern. The second pattern is phi_1 2 phi_2 2 … phi_h 2. The M th pattern is phi_1 M, phi_2 M and so on up to phi_h M. This is my phi and correspondingly, each one is multiplied with W.

(Refer Slide Time: 29:52)



Let me not put it here. This I put here. This is my big phi (Refer Slide Time: 29:57). I multiply this with w_1, w_2, … w_h, because they are fixed. The weights are fixed. The pseudo-inverse technique is offline training (Refer Slide Time: 30:22). It will allow the patterns to pass through the network, compute the y_d, store them for M patterns and while doing all these things, the weights are all fixed. Hence, what we are doing is that we are keeping all these relation vectors which is phi_1 to phi_2, this phi vector, and we are equating with the output. Obviously, if we multiply phi_1 1, phi_2 1 up to phi_h 1 with this weight vector, the corresponding output is y_d 1 pattern. The second one is y_d 2 and the m the one is y_d M. What we have got is a matrix notation. This is phi, this is weight and this is Y. Now, you can easily see that because the output has a linear relationship with weight, we can always represent in terms of matrix format and you can easily see that Y is an m by 1 vector, W is an h by 1 vector and phi is an m by h vector.

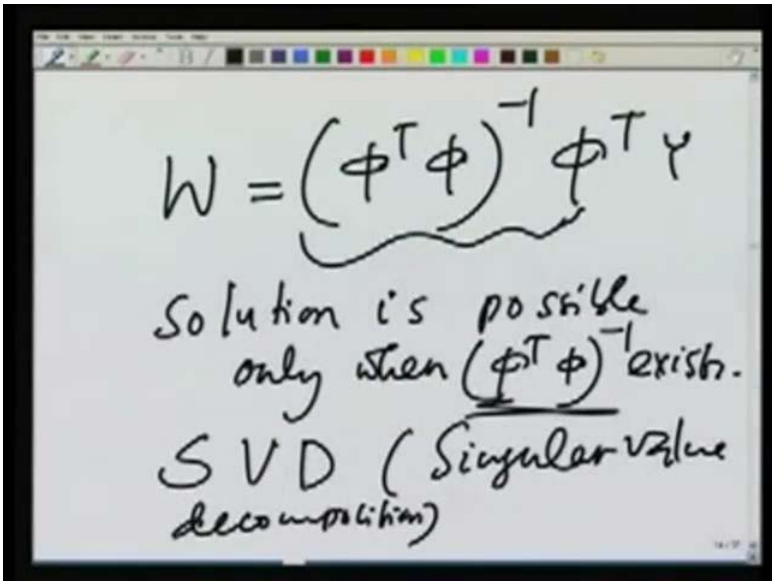
(Refer Slide Time: 32:02)



The solution will be that capital phi into W is Y. We found out this is M by h, h by 1 and this is M by 1. Since this is not a square matrix, we cannot invert it. The solution is very easy to find out when phi is a square matrix and is invertible but when it is not a square matrix, what we can do is we can use the pseudo-inverse technique. What is pseudo inverse? I multiply phi transpose phi W. phi transpose is h into M cross M cross h. So, this is an h cross h matrix and W is h cross 1. Here, you multiply phi transpose Y (Refer Slide Time: 33:19). This is h M M 1 and this is

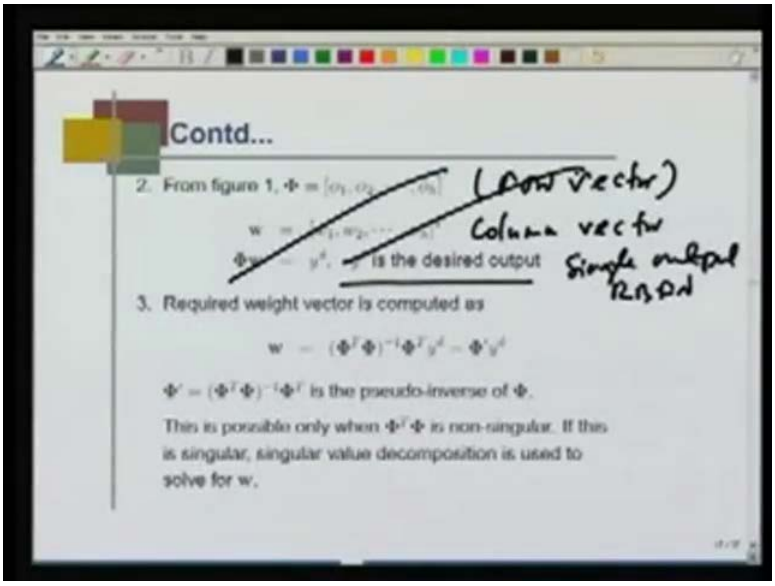
again h cross 1. Everything is satisfied. Finally, W is (phi transpose phi) inverse into phi transpose Y because this is now a square matrix but this solution is possible only when….

(Refer Slide Time: 33:55)



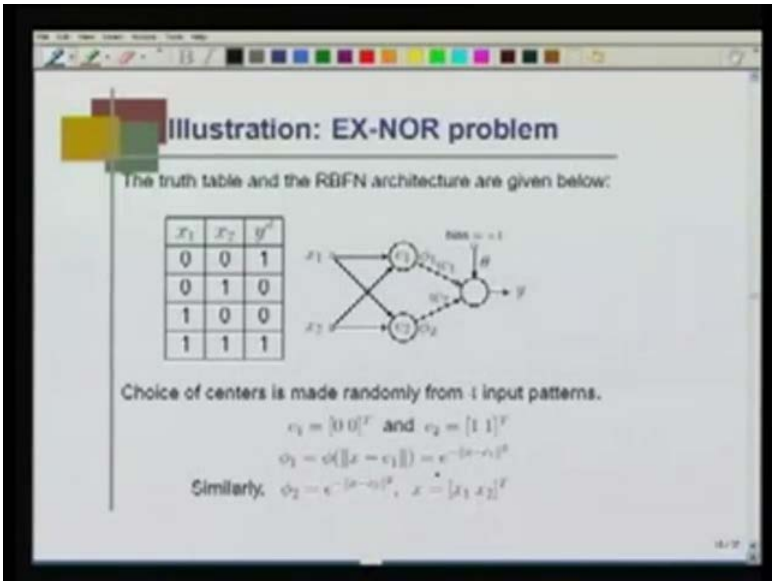
The solution we got for W is now (phi transpose phi) inverse phi transpose Y. This is known as pseudo…. The solution is possible only when (phi transpose phi) inverse exists. Even otherwise, we can use the singular value decomposition method when this is singular; singular means its determinant is 0.

(Refer Slide Time: 35:07)



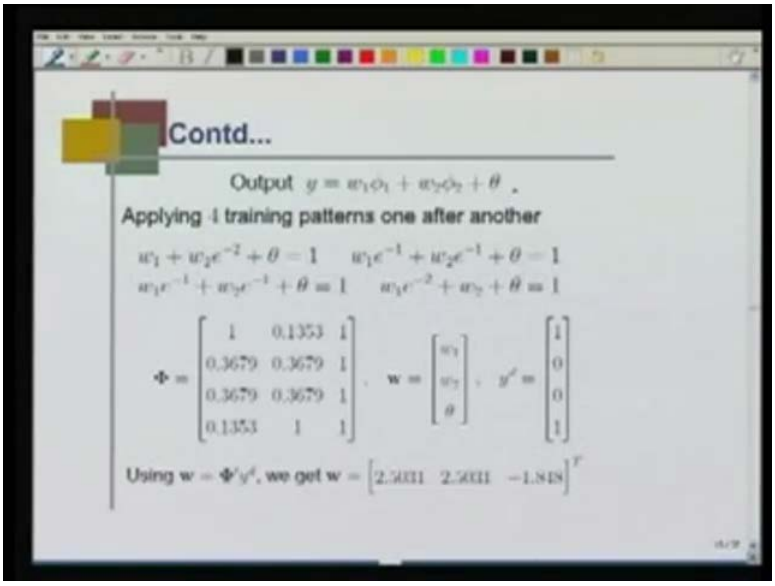
There was a little mistake there and I corrected that. What you found out is that the required weight vector w is (phi transpose phi) inverse phi transpose y d and this is known as the pseudo-inverse of phi. This is possible only when phi transpose phi is non-singular. If this is singular, singular value decomposition used to solve for w.

(Refer Slide Time: 35:29)



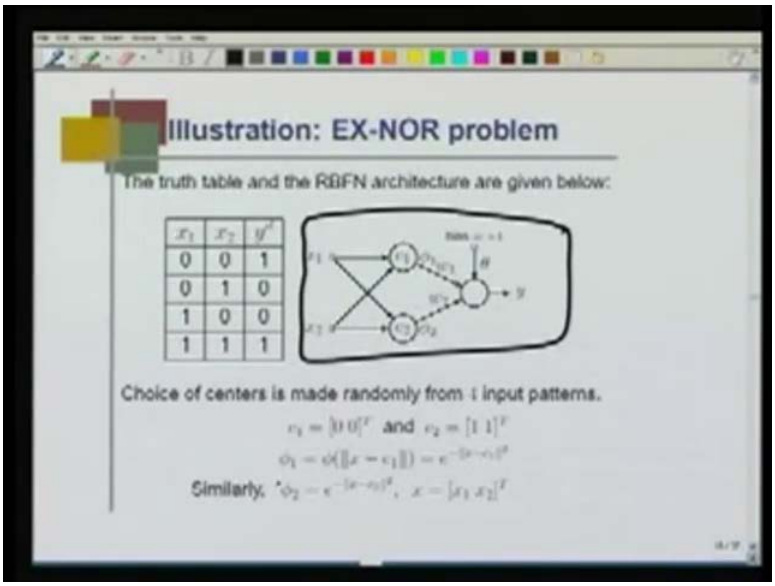
This is an example. We take an EX-NOR function: 0 0 1, 0 1 0, 1 0 0, 1 1 1. This is an EX-NOR function. You select two centers. Centers can be selected randomly from the input space. In input space, we only have four data and we can select any one of them. Let us select 0 0 and 1 1 as the centers. c_1 = 0 0 is one of the centers and the other center is 1 1. So, phi_1 is exponential minus (x minus c_1) square and phi_2 is e to the power of (x minus c_2), Euclidean (Refer Slide Time: 36:15) square, where x is x_1 and x_2.

(Refer Slide Time: 36:25)



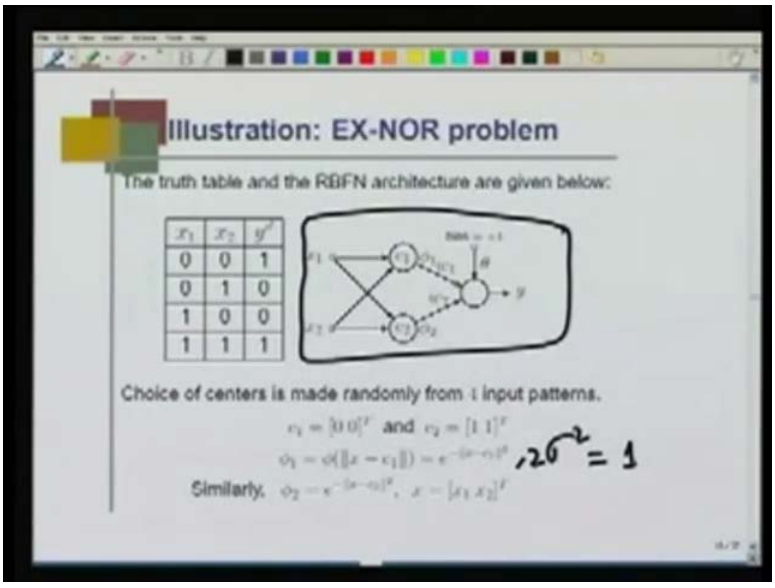
The output y is w_1 phi_1 plus w_2 phi_2 plus theta.

(Refer Slide Time: 36:31)



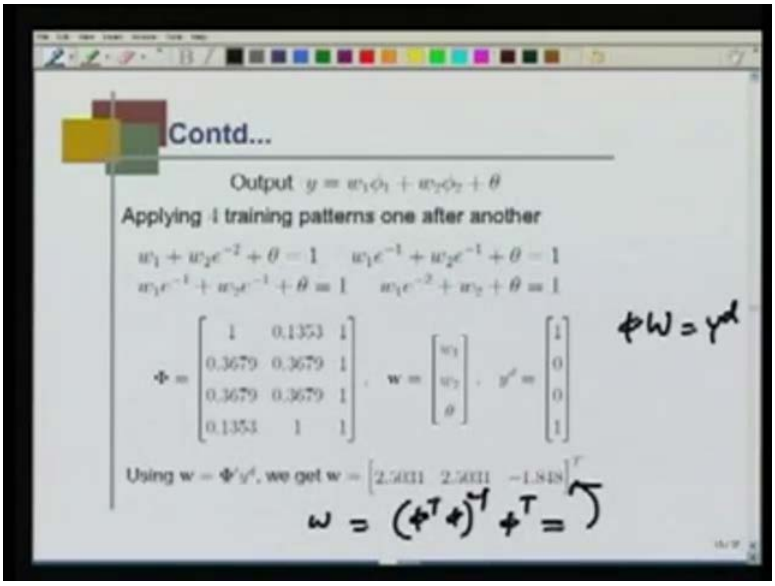
If you look here, theta is the bias. This is the architecture we selected. This is radial basis function network architecture selected to learn the function EX-NOR. We have taken two radial centers and one bias (Refer Slide Time: 36:51). The bias input is +1, the bias weight is theta, y is w_1 phi_1 plus w_2 phi_2 plus theta and phi_1 phi_2 are computed according to this.

(Refer Slide Time: 37:07)



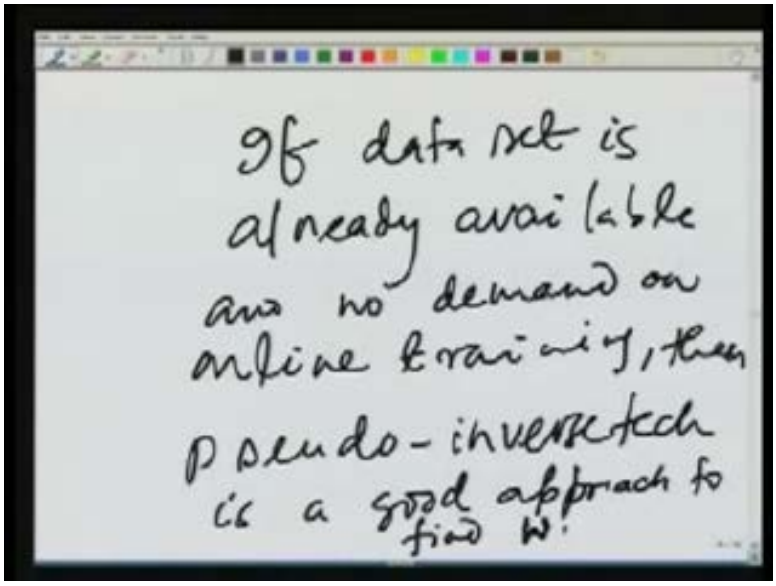
We have four training patterns. If we compute for each pattern the phi_1, the phi_1 for the first pattern is 1 (Refer Slide Time: 37:23) because you can easily see here for the first pattern 0 0 and c_1 is 0 0. Obviously, e to the power this Euclidean distance is 0 (Refer Slide Time: 37:32), so e to the power of 0 is 1. That is what you are saying. w_1 into phi_1, phi_1 is 0 here (Refer Slide Time: 37:39) and phi_2 is e to the power of minus 2, because the Euclidean distance is root 2 and square is 2, so e to the power minus 2. To let you know, we have only taken e to the power Euclidean distance square; that means we have already selected sigma equal to 1. It implies that 2 sigma square is 1 in the Gaussian function.

(Refer Slide Time: 38:12)



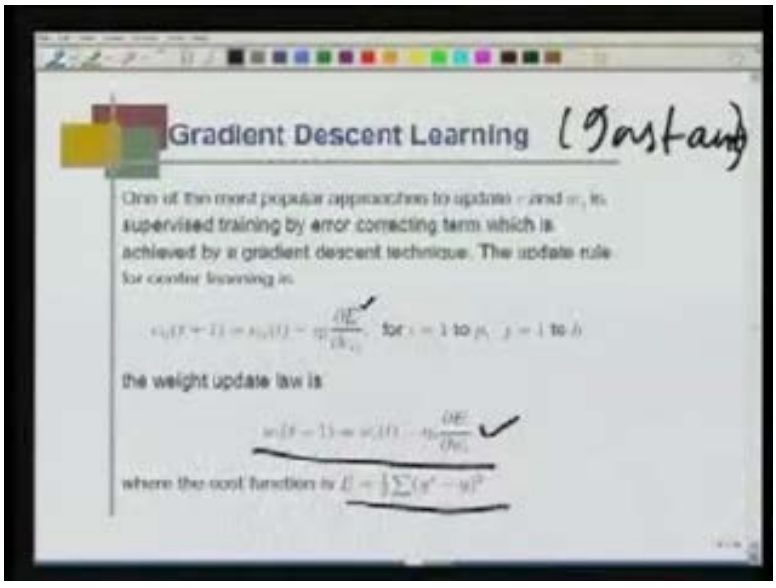
Based on that for each pattern…. This is for 0 0 (Refer Slide Time: 38:16), this is the next pattern 0 1, 1 0 and 1 1. Obviously, for 1 1, phi_2 is 1. These are the four equations we have for four data patterns. If you form a matrix phi into w equal to y d, phi W is Y d. If you put this, then y d is (1, 0, 0, 1), that is, y d is (1, 0, 0, 1) (Refer Slide Time: 38:53) and this is your phi and w. w is (w_1 , w_2, theta) – we have two weights and one bias. If we solve using pseudo-inverse technique, where w is (phi transpose phi) inverse phi transpose, this is this value (2.5, 2.5, and —1.8), which is a column vector. This is the solution. To conclude, if we have already some set of input/output data, we have no restriction for online training, then pseudo-inverse technique is good.

(Refer Slide Time: 40:05)



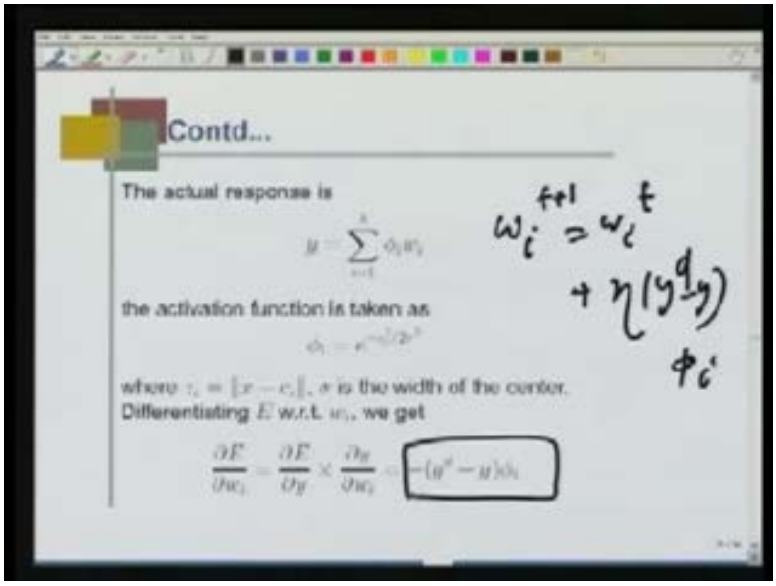
If a data set is already available and no demand on online training, then pseudo-inverse technique is a good approach to find the weight vector W. This is our weight vector. We talked about the pseudo-inverse technique where we fix the radial center, but imagine the situation where the data is coming online and you have to do online training. For example, in a control system, when we do a control system, data is coming online and we have to train our controller online. In that case, how should the training be done? Gradient descent as usual.

(Refer Slide Time: 41:35)



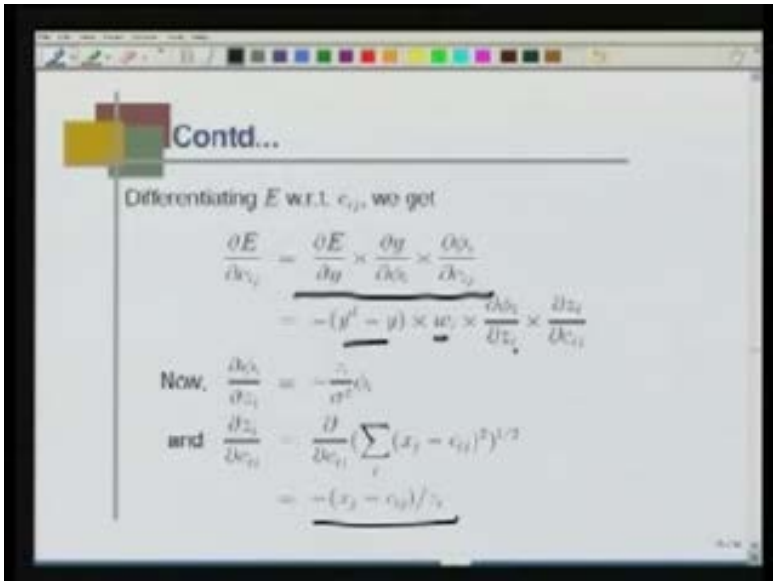
Instantaneous gradient decent is a good method for it. This is again the same methodology –once you are given instantaneous cost function E, c_{ij} t plus 1 is c_{ij} of t minus eta_1 del E upon del c_{ij} and this can be easily updated. Similarly, for weight also, we can do another gradient descent, where (Refer Slide Time: 42:01) the instantaneous cost function y d minus y. These are instantaneous values; these are all scalars, not vectors.

(Refer Slide Time: 42:15)



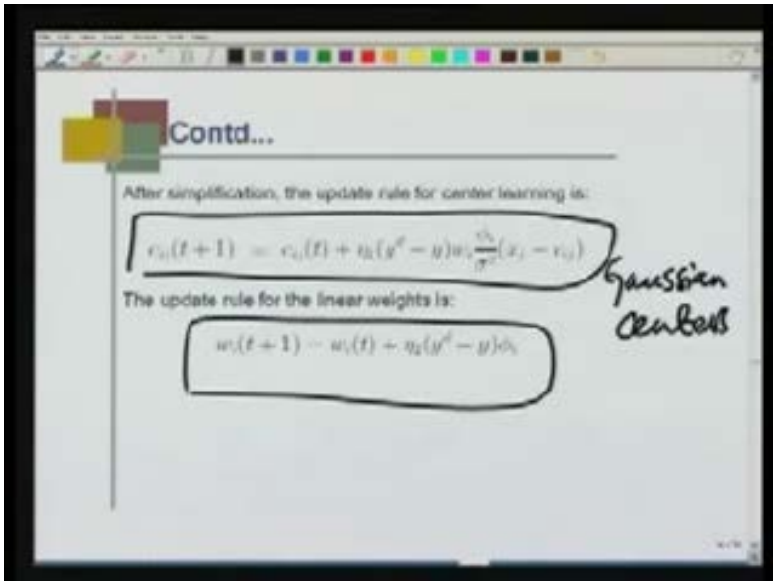
Just to give an example, I will be little faster here because we have already discussed a lot about gradient descent derivation. Our y is phi_i w_i. phi_i is e to the power minus z_i square upon 2 sigma square, where z_i is x minus e_i – the Euclidean distance between x and c_i, and sigma is the width of the center. Differentiating E with respect to w_i, you get this particular thing (Refer Slide Time: 42:46). Obviously, the weight update is w_i t plus 1 is w_i t plus eta into error y d minus y into phi_i. This is the weight update for weights in a radial basis function network. It is very simple because it is a linear network.

(Refer Slide Time: 43:18)



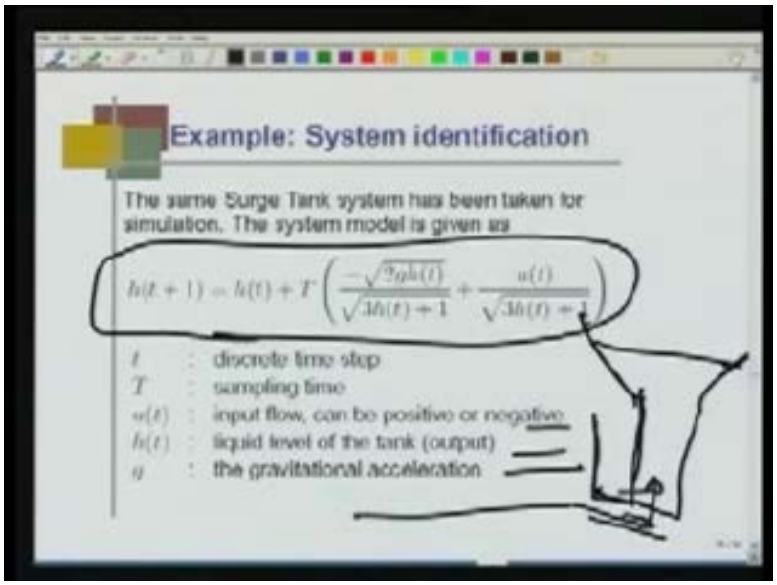
The weight update of center…. Imagine each center has p elements because x, the input, is a p-dimensional vector. That is why if you look at the derivation del E upon del c_i, this is this particular thing (Refer Slide Time: 43:40). First, you differentiate E with respect to y, y with respect to phi_i and phi_i is a function of c_{ij}. c_{ij} is not there in any other radial; only the i th radial center contains the element c_{ij} – this is important and hence this expression is right. We already know del E upon del y is this (Refer Slide Time: 44:07), del y upon del phi_i was w_i and to compute phi_i by this, we differentiate phi_i with respect to z_i, z_i with respect to c_{ij}. When you differentiate phi_i with respect to z_i, we get z_i upon sigma square phi_i negative and similarly, for z_i with respect to c_{ij}, we get this expression. This is negative, this is also negative, this is also negative, so overall, the sign of this expression is negative.

(Refer Slide Time: 44:43)



c_{ij} t plus 1 is thus our final expression for Gaussian centers and this is for weights. This is a simple derivation and you can verify for yourselves.

(Refer Slide Time: 45:12)

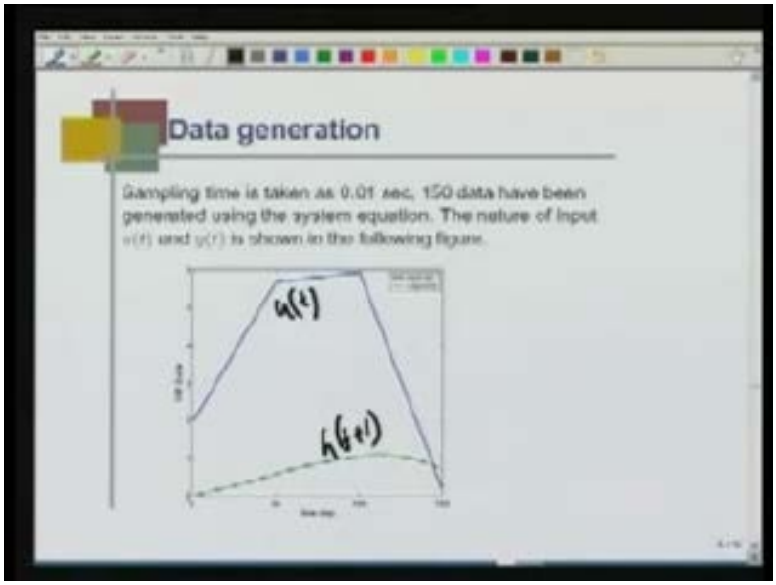


Using this online training – instantaneous gradient descent, we will now do a system identification of a surge tank. We have already discussed what a surge tank is. A surge tank is there to minimize the effect due to sudden pressure in the water reservoir. Normally, in a hydro

power plant, we have a water reservoir and suddenly, the level increases. From the water reservoir, there is a connection to the turbine. If a sudden increase is there, then the flow increases. To maintain the same flow, we place a surge tank here. The flow goes up and the liquid level in the tank increases. In the surge tank, the volume has a nonlinear relationship with the level h. Then, one of the models of such a surge tank is this. It is a nonlinear model h t plus 1 is equal to h t plus T into this quantity (Refer Slide Time 46:38), where h t is the liquid level and u t is the flow inside this surge tank.

What you are seeing here.… Let me do it like this. This is our water reservoir, this is the surge tank and we are only concerned with the model of this surge tank. This model of the surge tank is that if there is a certain flow rate into the tank, then how the level of the tank increases. For a nonlinear thing, this is the… for a discrete dynamic model of the surge tank. Please see this – T into minus square root of 2 g h t upon square root of (3 h t plus 1) plus u t upon square root of 3 h t plus 1, where u t is the water flow into the surge tank, h t is the liquid level, g is the acceleration due to gravity, T is the sampling time (Refer Slide Time: 47:43) and t is the sampling instant.

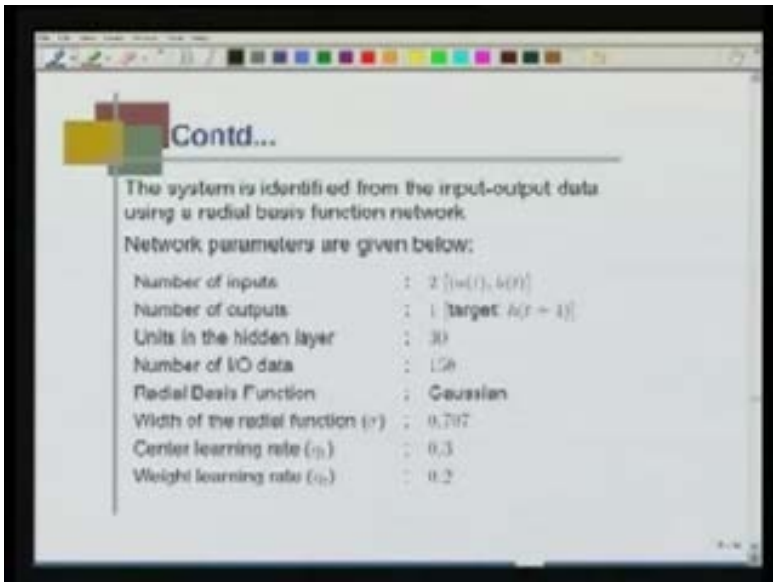
(Refer Slide Time: 47:48)



We have generated data in a similar manner –we have done it earlier. Sampling time is 0.01 second and 150 data have been generated using this data. Input flow is according to this

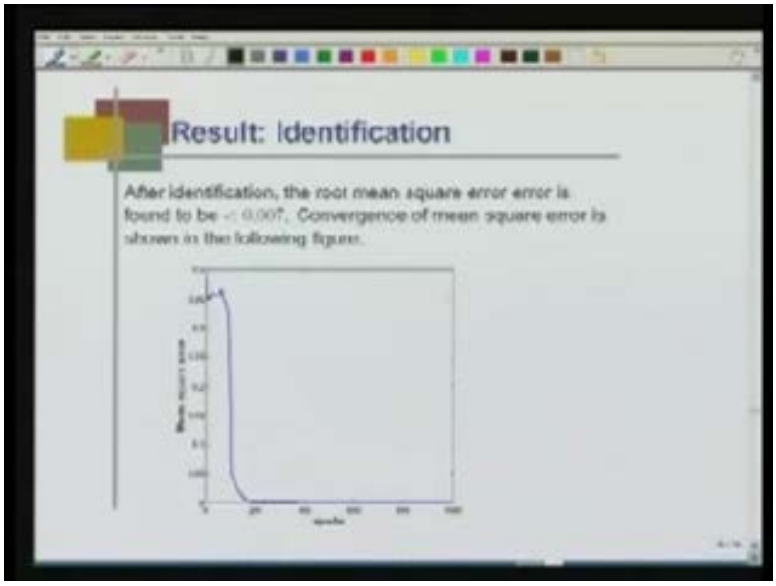
particular curve and corresponding h t liquid level. This is your u t and this is your corresponding liquid level.

(Refer Slide Time: 48:17)



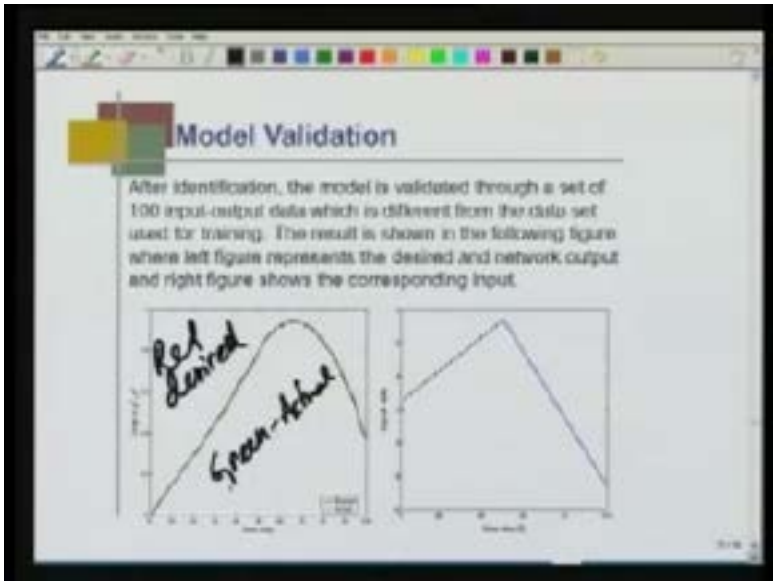
We selected a radial basis function network. Obviously, it has two inputs and one target. The two inputs are u t and h t, the target is h t plus 1, the units in hidden layer are 30, numbers of input/output data is 150, the radial basis function is Gaussian, the width of the radial function sigma is 0.707, the center learning rate eta_1 is 0.3 and the weight learning rate is 0.2.

(Refer Slide Time: 48:46)



You can easily see that this is the convergence. Within 200 or less than 200 epochs, the convergence is achieved to achieve the root mean square error below 0.007.

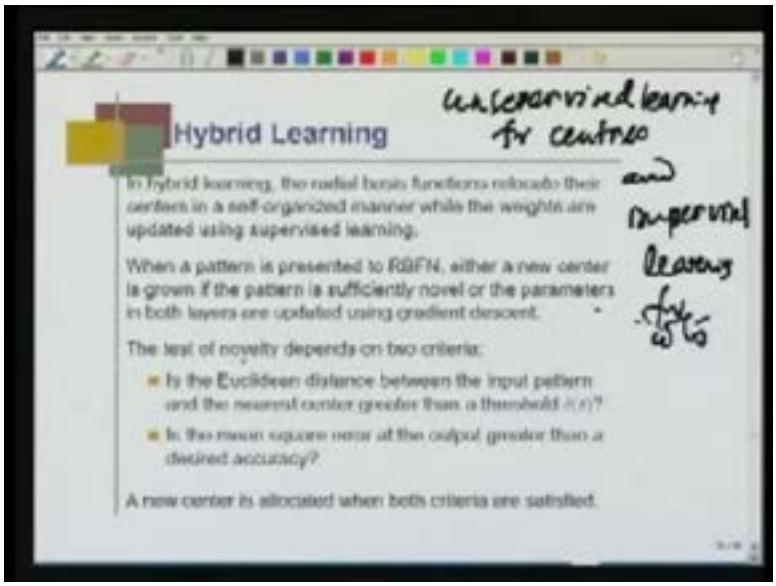
(Refer Slide Time: 49:02)



When we give a new input, the input flow follows this particular curve (Refer Slide Time: 49:09), then the liquid level here. You can see the red curve and the green curve. Over the red curve, there is a green curve and red is desired and green is actual. You can easily see that the

RBFN model that has been trained, which is 200 epochs, could easily map or could easily learn the dynamic of the surge tank.

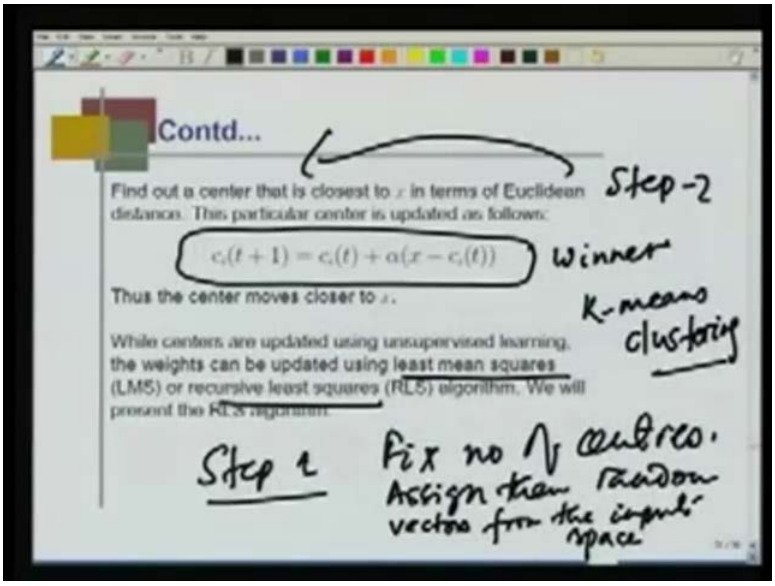
(Refer Slide Time: 49:55)



We talked about two different learning: pseudo-inverse and gradient descent. Now, we will talk about a new kind of learning that is normally employed for radial basis function network, which is called hybrid learning. What is hybrid learning? Center and weights are separated. Centers, since they represent the clusters in the input space, we can use unsupervised learning to learn the centers, whereas it is supervised learning for the weights. Hybrid learning means unsupervised learning for centers and supervised learning for weights. In hybrid learning, the radial basis functions relocate their centers in a self-organized manner, that is, unsupervised learning, while the weights are updated using supervised learning.

When a pattern is presented to RBFN, either a new center is grown if the pattern is sufficiently novel or the parameters in both layers are updated using gradient descent. The test of novelty depends on two criteria: Is the Euclidean distance between the input pattern and the nearest center greater than a threshold? Is the mean square error at the output greater than a desired accuracy? A new center is allocated when both criteria are satisfied.

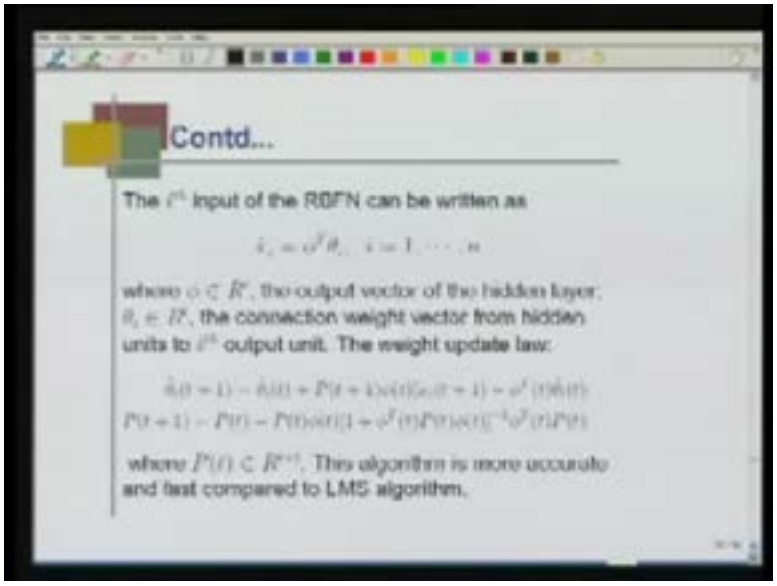
(Refer Slide Time: 51:32)



Normally, the easiest way to do this center learning using the classical clustering is that we fix the number of centers and assign them random vectors from the input space – this is the first step, this is step 1. This is step 2. In step 2, what will do is once the centers are fixed, uniformly sampled from the input space, then what we do is we present an input pattern, find the Euclidean distance between this input pattern and all these centers which are already fixed, the numbers are also fixed, and then find the winner. Whichever is the winner, you update the weight of that winner. This is for the winner and for all other centers, we do not do any changes, the centers remain as is.

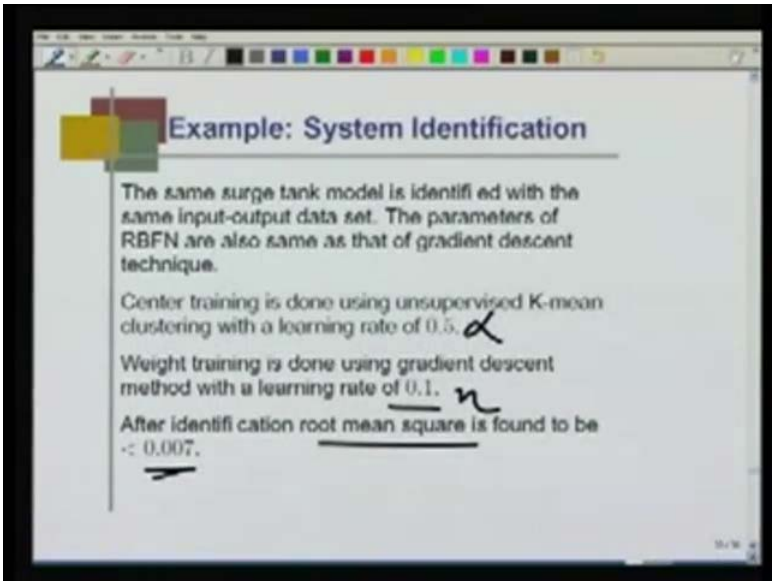
We repeat this process for all the data patterns from the training set. This is called K-means clustering. Not only can radial centers use K-mean clustering, but there are other clustering techniques. We will not focus on that now, we will just give an idea how this is done – clustering, unsupervised manner. What about the weights? Weights can be used because weight and the output have a linear relationship. We can use any least mean square, we can also use gradient descent. Apart from gradient descent, we can also use the least mean square algorithm or recursive least square algorithm.

(Refer Slide Time: 54:05)



This is a recursive least square algorithm. I will not discuss this in this class, but maybe later.

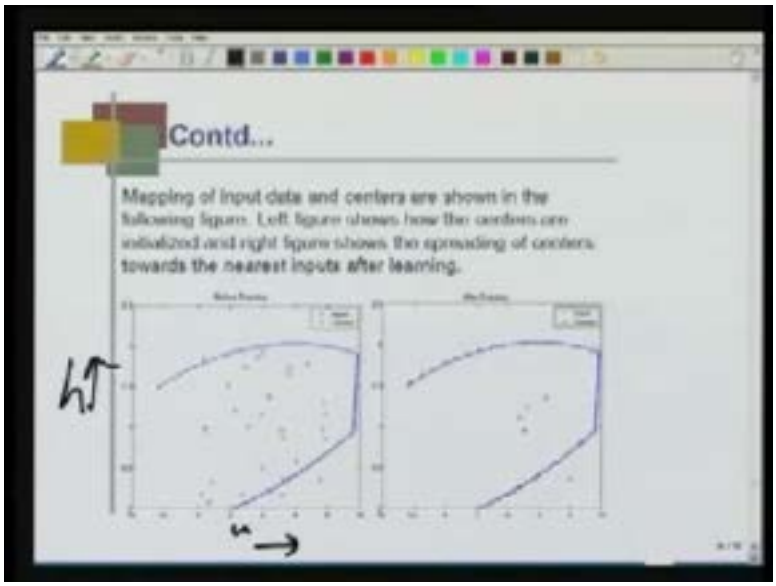
(Refer Slide Time: 54:12)



The surge tank can also be modeled using this hybrid learning, where K-means clustering has been used with a learning rate 0.5, which is alpha. Alpha is 0.5 (Refer Slide Time: 54:36). The gradient descent method has been used for weight update, where the eta is… this is eta, this is

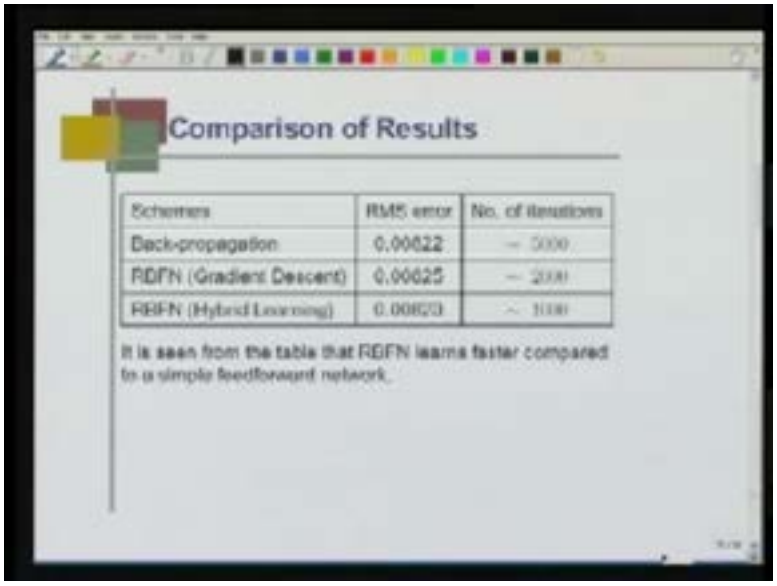
alpha (Refer Slide Time: 54:45) and the training is terminated when root mean square error was less than 0.007.

(Refer Slide Time: 54:56)



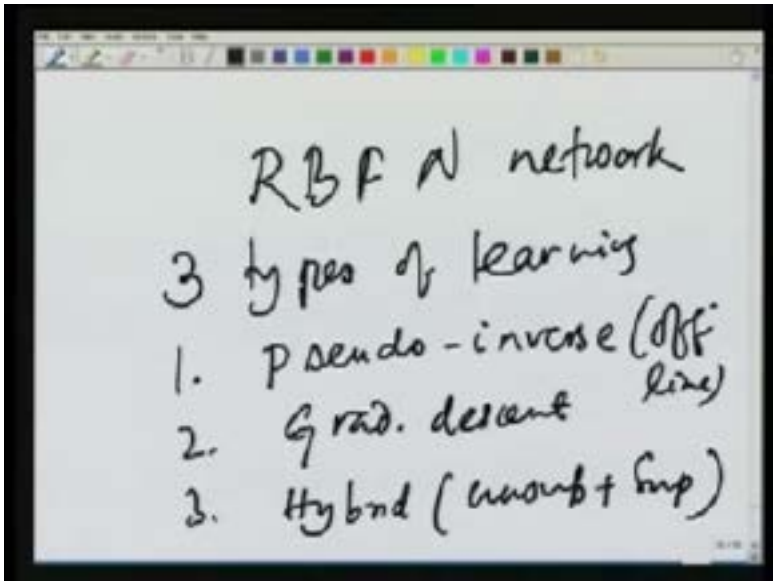
You can easily see that in the beginning, when we reshuffled this, the centers were uniformly distributed. These are the centers. The circles are the centers and this is my input data (Refer Slide Time: 55:16). This is my u and this is my h. What you are seeing is that before unsupervised learning, the centers are all randomly distributed – this is the data and as we presented data to the centers using K-means clustering, you see that most of these centers were aligned with the input data and very few are left unaligned.

(Refer Slide Time: 56:05)



This is the final comparison of the result. What we are seeing is that when we train a surge tank, do the system identification of a surge tank using back propagation network, that is, multilayer network, the radial basis function network using gradient descent, radial basis function network (hybrid learning), then the number of iterations that are required… you see that back propagation takes a long time (Refer Slide Time: 56:35) and the radial basis function network takes less time. The RMS error is the same because we have fixed the RMS error – the training is terminated over the same RMS error. Obviously, the same RMS error for the new data… that means generalization for all the three networks are the same.

(Refer Slide Time: 57:11)



The conclusion is that we discussed today RBFN network and we talked about three types of learning. The first one is pseudo-inverse, which is offline training, the second is normal gradient descent and the third one is hybrid, which is a combination of unsupervised plus supervised. Thank you very much.