Course Name: Machine Learning and Deep learning - Fundamentals and Applications

Professor Name: Prof. M. K. Bhuyan

Department Name: Electronics and Electrical Engineering

Institute Name: Indian Institute of Technology, Guwahati

Week-10

Lecture-36

Welcome to NPTEL online course on machine learning and deep learning fundamentals and applications. In my last class, I explained the concept of the radial basis function. Today, I am going to explain the concept of the supervised and the unsupervised artificial neural networks. And mainly I will be considering one popular learning algorithm that is the back propagation learning algorithm. In the back propagation learning algorithm, we know what is the desired output and also we can calculate the actual output. The difference between the desired output and the actual output that is the error, the error is back propagated to the input to adjust the weights of the artificial neural network.

So knowledge of the artificial neural networks is available in the form of weight vector. So that concept I will be explaining that is the concept of the supervised learning. And after this, I will be explaining the concept of unsupervised clustering by considering some popular neural networks. One is the competitive learning and another very popular algorithm is kohonen neural networks.

So let us begin this class. So the first point is the training. So training for the supervised artificial neural network. So training means finding appropriate weights and other parameters of the artificial neural network. In my last class, I mentioned that the knowledge of the artificial neural network is available in the weights.

So during the training, I have to determine the weights and other parameters of the artificial neural networks. And that is the concept of the supervised artificial neural network. So that I will be explaining in detail in this class. And some basic neural network structures, so I am explaining some of these networks. The first one is the multilayer feed forward network that is the multilayer perceptron MLP.

Another network is the feedback or recurrent network, Hopfield network, competitive network, self organizing network. So briefly I will explain the concept of all these networks and mainly the concept of the MLP, the multilayer perceptron and also the

concept of the competitive network and the self organizing network. So what is MLP? Move to the next slide. In case of the multilayer feed forward network or the multilayer perceptron MLP, you can see in the figure I have one input layer, one output layer and in between one hidden layer I am showing. There may be more than one hidden layers, maybe two hidden layers, three hidden layers based on the complexity of the problem.

For a simple problem, I may only consider input layer and the output layer. No need of considering the hidden layer. If the samples of the classes are linearly separable and if I consider a very simple classification problem, then I can consider only simple artificial neural network without hidden layer. That means I have only input layer and the output layer and there is no hidden layer between the input layer and the output layer. But for a complex problem I have to consider hidden layers between input layer and the output layer.

And in the figure you can see I am showing the nodes corresponding to the input layer, the nodes corresponding to the hidden layer and the output node also I have shown in the figure. So this is one example of the multilayer feed forward network or the multilayer perceptron MLP. So move to the next slide. Here also I have shown the MLP that is the feed forward network. So we have one hidden layer between the input layer and the output layer.

So you can see the input neurons and the output neurons. And in between the hidden layer is there. So you can see the hidden neurons also and also the interconnections between the neurons. So the connecting weights if I consider suppose the connecting weight is W_{ij}. So this is the connecting weight between the input node and the hidden nodes.

So all the possible connections I am considering here. And similarly we may consider the weight between the hidden layer and the output layer suppose W_{kl}. So this is the weight. So all these weights I have to consider. So this is one example of the feed forward neural network.

Now why actually we need the hidden layer. So already I mentioned that for a complex pattern classification problem we have to consider the hidden layer. If I if the problem is very simple then I may consider only the input layer and the output layer but for a complex pattern classification problem I have to consider hidden layers. There may be single hidden layer only one layer or maybe multiple hidden layers between input and output layers. So why actually we need the hidden layers.

So corresponding to multi-class classification problem I can give one example multi-class classification. So this problem I am considering and suppose I have three

classes ω_1, ω_2, ω_3. So three classes I am considering and corresponding to this I am considering one feed forward network. So suppose this is X1 X2 Xd. So we are considering a d-dimensional feature vector the feature vector is X and we are considering a hidden layer suppose this is h1 h2 h3 so these are the nodes of the hidden layer and also we are considering the output layer.

So these are the nodes of the output layer and we can consider all the possible connections we can consider the connections like this that is the interconnections between the nodes of the input layer and the hidden layer. So these connections I have to determine so like this I may have this type of connections I am not showing all the connections. So this feed forward network I am considering. So the input is X and the hidden neurons are h1 h2 h3 and we have the output. So the input is y1 y2 y3.

So for this classification problem suppose this is my these are my decision boundaries. So suppose this parameter is H1 H2 and H3. So I may consider like this 0 1 0 1 and 0 1. So suppose if I consider this region that corresponds to the class ω_1. So this is the class ω_2 this portion and this region corresponding to the class ω_3.

So suppose I am making a table h1 h2 h3. So suppose h1 is 1 and h2 is also 1. So here you can see in this figure the h1 is 1 and h2 is 1. So that means output what output I have to consider output will be y1 should be equal to 1 and that corresponds to the class ω_1. So in this case h1 is 1 and h2 is 1 and h3 that is the don't care.

We are not considering this h3 it may be 0 or 1 that is not important but h1 should be 1 and h2 should be 1 and corresponding to this y1 should be equal to 1 and that is nothing but the class ω_1. So that means we are considering this region that is the ω_1 we are considering that class we are considering. And suppose if I consider h2 is 0 and h3 is 1. So that means we are considering this region this 0 we are considering and 1 is considered h2 is 0 and h3 is 1. So corresponding to this the class is y2.

So y2 should be the output. So output y2 will be 1 and that corresponds to the class ω_2. And in this case this h1 is don't care. So we are not considering h1 but the important is h2 and h3 we are considering. So that means we are considering the class ω_2.

So if the h2 is 0 and h3 is 1 then corresponding to this the class will be ω_2 and corresponding to this y2 will be equal to 1. And finally if I consider h1 is 0 h2 is don't care and h3 is also 0. So that means we are considering this one 0 0 and that corresponds to the class ω_3. So in this case the output y3 should be equal to 1 and that corresponds to the class ω_3. So you can see the importance of the hidden neurons.

So hidden neurons we are considering h1, h2, and h3 and based on these values I can get

the outputs. So outputs are y1, y2, and y3. So from this explanation you can understand the importance of the hidden layer. So this problem is the multi-class classification problem and we are considering three classes ω_1, ω_2, and ω_3. So next one is the feedback network that is also called the recurrent network.

In case of the feedback network the output is fed back to the input layer. So that is the definition of the feedback network. So after this the Hopfield network. So in case of the Hopfield network every node is connected to every other node. So here in the figure you can see the node 1 is connected to 3, 3 is connected to 2, 2 is connected to 1.

So that means that every node is connected to every other nodes and also we are considering the symmetric width. So what is the symmetric width? So suppose if I consider the connections between 1 and 3 so from 1 to 3 the weight is - 2 and from 3 to 1 the weight is - 2 that is the symmetric width. Similarly if I see the connections from 3 to 2 so from 3 to 2 the weight is 1 and from 2 to 3 it is 1. So that is called a symmetric weight. So every node is connected to every other node and we are considering symmetric weights and that is the one definition of the Hopfield network.

So after this there is another important network that is the competitive network. So this is actually employed for unsupervised clustering. So in the competitive network so first part is if you see that first part is the feed forward network. So from here to here so this is nothing but the feed forward network and we are considering a competitive layer. So the competitive layer is nothing but a comparator.

So that is nothing but the comparator. So the input to the competitive layers suppose the input is I1, I2, I3 so actually these are the outputs of the feed forward network and these are the inputs to the competitive layer and the outputs are suppose O1, O2, O3 so these are the outputs of the competitive layer. So in a particular instant what happens suppose I2 is maximum so there is a comparison between I1, I2 and I3 so which one is maximum at a particular time. So corresponding to that condition the output will fire that means I2 is maximum so I am getting the output O2 I am getting. So I2 is maximum so corresponding to this the output will be O2 so at a particular time I will get the output O2.

So there is a comparison between I1, I2 and I3 out of which one is maximum that is considered and corresponding to this we get the output. So that is the objective of the competitive layer. So this is one architecture of the competitive layer that is the competitive layer architectures. In this case we are considering the inputs D1, D2, and D3 so these are my inputs so outputs are O1, O2 and O3 so these are the outputs and I want to determine the minimum which one is minimum D1 is minimum or D2 is

minimum or D3 is minimum. So if D1 is minimum then O1 will be 1 so suppose that D1 is minimum corresponding to this O1 will be equal to 1 otherwise it will be 0.

So if I consider this competitive layer architecture for determining the D1 as minimum I am considering this structure inside the box and we are considering two threshold threshold 0 and threshold $\frac{3}{2}$. So you can also determine this condition so randomly you can select some values suppose this value is supposed to 3 + 5 and just you apply this principle here you can determine the neuron outputs you can see corresponding to D1 is minimum O1 will be equal to 1 so you will get that one. So this is a very simple architecture for the competitive layer and we can determine the minimum out of these three D1, D2 and D3 and after this the concept of the self organizing network. So in the self organizing network you can see this is the structure and this is also called the Kohonen neural network. So this is the self organizing map that is the SOM self organizing map the Kohonen neural network.

So you can see this is the input vector input vector is x, x is the input vector so n dimensional vector and these vectors are mapped into a two dimensional space. So if you see the definition of this self organizing network so a mapping is considered from the input n dimensional data space onto one or two dimensional array of nodes. So in this figure we are considering a two dimensional array the size is x × y so this is x and this is y so two dimensional array is considered and we are doing the mapping mapping of the input vector and corresponding to this we have the nodes in the array. And one important point is topological relationships in the input space are maintained. So that means here you can see this suppose x1 is close to x2 in the two dimensional space that is in the array that is maintained.

So x1 should be close to x2 so that topological relationship is maintained in the array that is the two dimensional array. So suppose this is this node is x1 and you can expect x2 or x3 near to this x1. So that is the topological relationships in the 2D array. So this the self organizing map this is employed for unsupervised clustering. So before going to that first let us discuss about the supervised learning that is how to train the supervised network that is the artificial neural network we can consider the feed forward network or the MLP that is the multi layer perceptron and how to do the training.

So in my last class I explained the concept of the training here again I am explaining the concept. So how to do the training with the help of the principle or the algorithm that is the back propagation training. So if you see this algorithm suppose I have a network and suppose the input is x and we are considering the weight vector is W and corresponding to this output is y. So what I have to do in this training that is the back propagation training first I have to apply inputs to the network. So I am applying the

inputs to the network and corresponding to this we can determine the neuron outputs we can determine.

So in case of the supervised learning we know what is the desired output and in this calculation we can determine the actual output. The difference between the desired output and the actual output that is called the error. So the error is back propagated to the input to minimize the error and because of this minimization I am adjusting the weights of the artificial neural network. So here you can see first I am applying the input to the network after this we are determining all the neuron outputs compare all outputs at output layer with desired outputs because we know the desired output and that is the concept of the supervised learning. So corresponding to x a particular x we know what is the desired output.

After this compute and propagate the error the error is nothing but the difference between the desired output and the actual output. So compute and propagate error measure backward through the network and minimize error. So we have to minimize the error and for minimization of the error we are adjusting the weights of the artificial neural network. So like this we have to do the training. So this process I have to do iteratively until a particular condition is not satisfied that is the convergence condition.

So here I am showing how to do the training that is the back propagation training. So we have the inputs x1, x2, xn. So we are considering n number of input nodes and W_{ij} that is the weight connecting input node i and the output node j. So weight is considered that is the W_{ij} we are considering that is the connecting weight between the input node i and the output node j. And we know what is the desired output at a particular node j.

So dj is the desired output at the node j and corresponding to this we can determine the error that is a square error. So square error we can determine that is the desired output and what is the actual output actual output is computed like this. This is nothing but the multiplication of the weight and the input and for all the neurons i is equal to 1 to n we have to do this. And we have to minimize the error so for minimization of the error we have to adjust the weights of the artificial neural networks. So that is why we are differentiating the error with respect to the weight the weight is W_{ij}.

So this differentiation I am doing and it is equating to 0 and what is the weight updation rule. Here you can see this is the new weight this is the old weight and this η is the learning rate. So it determines the learning rate and this term is nothing but the difference between the desired output and the actual output. So this is the weight updation rule and this is nothing but the gradient descent algorithm and that already I have explained in one of my class. So it is nothing but a gradient so this is the gradient descent algorithm.

So we have to adjust the weights of the artificial neural network so that we can reduce the error and this learning rate actually it determines the convergence. If η is small the convergence will take time and if the η is very high the convergence I will get immediately but that may not be so accurate. So there should be a compromise between the high η and the low η value. So this is the concept of the back propagation algorithm.

So in my next slide I will explain this concept again. So here you can see this is the feed forward network we are considering and we have the input nodes and you can see I am considering the input is x1, x2, x3. So this is the input layer and input feature vector is x and we are considering one bias input. Already I have explained the importance of the bias input and again you can see I am considering the second layer that is the hidden layer and in the hidden layer we are considering the one bias input. So this bias input we are considering and again in the third layer also you can see I am considering the bias input and these are the neurons x11, x21, x31 these are the neurons in the hidden layer one hidden layer and another neuron neurons are x12, x22, x32. So these are the neurons of the hidden layers and you can see the connecting weights.

So you can see all the interconnections and in between you have the weights. So suppose this weight is W_{ij} so we may consider like this. So during the training I have to adjust the weights of the artificial neural networks. So we have to adjust the weights. So we have to get the values of the weights so that we can reduce the error.

The error is nothing but the desired output minus actual output. This is the difference between the desired output and the actual output that is the error. So I have to minimize the error and I am applying the training samples and corresponding to the training samples we know what is the desired output and we can determine the actual output and based on this we can determine the error. The error is back propagated to the input to adjust the weights of the artificial neural network. So this is the concept of the back propagation training.

So the same thing here I am explaining here. So suppose simple network I am considering. So these are the input nodes and suppose x is the input vector, physical vector. So W is the weight vector and y is the output that is nothing but W transpose x. So I am getting the output like this. So what is the error? Error is nothing but the desired output minus actual output.

So y is the actual output. So we can determine the square error we can determine and we have to minimize the error with respect to the weight vector W. So $e^2 = -2(y_d - y)x$. So I will be getting this expression. So how to get this expression? Suppose y = W1x1 + W2x2 + …. like this.

This is the y. So, I can determine y. e^2 that is the squared error. So (y_d - W1 x1 - W2 x2 - …) like this. This is the squared error. So now I have to differentiate e^2 with respect to the weight W1.

So, differentiating the error with respect to W1. So if I do the differentiation, so you can see I am getting - 2 (y_d - W1 x1 - W2 x2 …) x1 I am getting. Similarly if I do the differentiation with respect to W2, in this case also I am getting - 2 (y_d - W1 x1 - W2 x2 …) x2. So I can do the differentiation like this. So actually, from this I am getting this one.

So from this actually I am getting this one. So what is the weight updation rule? So finally I have to consider the weight updation rule that already I have mentioned. So how to update the weight? So W star is the new weight. This is the old weight W is the new weight and we are considering $\frac{1}{2}\eta$, η is the learning rate and this the differentiation with respect to W that is the error square is differentiated. So I am getting the weight updation rule and this is actually the gradient descent algorithm.

So this is the concept of the back propagation training. So now move to the next slide. So what is the unsupervised learning? So unsupervised artificial neural networks can be employed for unsupervised clustering like the K Means clustering or the fuzzy K Means clustering we have already discussed the unsupervised artificial neural networks can be employed for clustering. So that concept I am going to explain. The first concept is the competitive learning. So in the neural network we are considering the competitive learning to generate the weight vectors that is nothing but the code vectors and we can consider this principle.

So first in the network I have to randomly initialize all the weights. After this the training vectors that is the vector x are applied one by one and if the output and if the output node J fires for a particular input x the corresponding weight vector is updated. So suppose if I consider one input is x and corresponding to this input if the output node J fires then the corresponding weight vector should be updated. So this is the updation rule. So WJ here you can see the WJ the old weight so this is the old weight and this is the new weight and this η already I have explained that is nothing but the learning rate that controls the learning rate x - WJ. So in this case we have to determine which one is the winning neuron corresponding to a particular input x I have to determine which one is the winning neuron and the winning neuron is updated by considering this updation rule.

So the basic principle I will explain in the next slide. So what is the competitive

learning? So suppose this is x1 like this up to xd so that is nothing but the input feature vector x that is the d-dimensional feature vector and we are considering the cluster centers y1, y2 these are the cluster centers and we are considering C number of cluster centers that means C number of clusters and we are considering the interconnections between the neurons. So all the connections we are considering and suppose corresponding to y1 my weight vector is W1 is a W1. So I have the clusters so this is suppose one cluster and the cluster center is y1 and like this I have C number of clusters so this is also another cluster and the cluster center is suppose yc, C number of clusters are available and we are considering one comparator. So, we are connecting to the comparators so all the connections we are not showing and the output terminals are 1 because we have C number of classes 2 C so C number of classes we are considering. So suppose this structure is considered so corresponding to y1 how to get y1 this you can see the input is x and with the help of the weight vector weight vector is W1 I am getting the output output is y1.

So in this case this actually yi particular yi depends on the weight vector wi that is the yi it depends on the weight vector wi because x is multiplied with W1 and I am getting y1 in this case I have to find a distance between x and the y's. So first I have to determine the distance between x and y1 x and y2 x and yc so I have to find all the distances and we have to find the minimum distance which one is the minimum distance. Suppose the distance between x and y1 that is minimum so I am getting the minimum distance corresponding to y1 and the corresponding weight vector is W1. So this W1 is the winner so suppose if I consider suppose this is yi so we are considering yi so this is the yi we are considering the distance between x and yi that is minimum and the corresponding weight is wi so I can write the distance between x and yi that is the minimum distance we are getting. So corresponding to this the weight is wi so this wi that is actually the winning neuron winning neuron.

So based on the minimum distance I can determine the winning neuron that is actually the winner wi is the winner. So how to update the winner the winner is updated like this so $w_i^* = w_i + \in(x - w_i)$ so the winner is updated like this. So this is the new weight this is the old weight $\in$ is some constant maybe small fraction we can consider maybe 0.8, 0.9 so it actually determines the learning rate so small fraction we are considering and this x - wi that is actually the learning vector x - wi is the learning vector.

So the winner is updated like this so now how actually we update the other neurons that means which are not winners. So suppose wj so this is not winner so corresponding to this the weight of this rule is wj. $w_j^* = w_j - \in \left(x - w_j\right) \forall\, i \neq j$. So this is the weight updation rule for all the losing neurons. So for winning neurons this is the weight updation rule that is for the winner and for other neurons this is the weight updation rule.

So you can see in one case it is plus in another case it is minus. So that means what I am doing so suppose this is my cluster and suppose this is my weight vector the weight vector is Wi and suppose this is my the vector is x so I am moving Wi to a new position the new position is w_i^*. So this is the new position so I am moving Wi towards x so that is the interpretation of this equation that is I am moving Wi towards x and what about the other neurons that is the Wj I am moving other neurons or other weights away from the feature vector x. So that means I am moving the weights corresponding to the winning neurons towards the feature vector x and I am moving other weight vectors away from the feature vector x. So that means I am giving more importance to the winners and because of this process I will be getting the clusters. So this process is very similar to the k-means clustering in the k-means clustering also we are finding the minimum distance between the input vector and the centroids.

In this case also I am finding the distance between the input vector x and the centroids. The centroids are y1, y2, yc so I have c number of classes or c number of clusters. So we are finding the minimum distance like the k-means clustering and based on this I am updating the weights. So that means I am giving more importance to the winning neurons. So always the winning neurons are updated.

So this problem is called the under-utilization problem. Because I am giving maximum importance to the winning neurons. So there is another variation of this competitive learning. So I am going to explain that one. So that is called the frequency sensitive competitive learning.

That is another form of the competitive learning. So that is the frequency sensitive competitive learning. So this is another form of the competitive learning. The weight updation rule is again like this. So w_i^* that is the new width the old width and we are considering the learning vector is like this x-Wi. So in this case this $\in$ so here actually it is the new width this is the old width and this $\in$ is now defined.

So $\in$ is now defined so it is $\frac{1}{Fi}$. So Fi is the frequency Fi is the frequency. Now what is the meaning of the frequency that means the number of times x is mapped to Wi. So how many times x is mapped to Wi. So Wi is the winning neuron. So based on this I can write $w_i^* = \in x + (1 - \in) w_i$.

So this is the weight updation rule. So this weight updation rule already I have explained that is in case of the competitive learning. Now I am considering this is the weight updation rule updation rule weight updation rule for the frequency sensitive

competitive learning. So this is the old width and this is the new width. So this w_i^* is now $\frac{1}{Fi}x + \frac{Fi-1}{Fi}w_i$.

I can write like this so it is nothing but $\frac{1}{Fi}[x + (Fi - 1)w_i]$. So that is w_i^* I can write like this. So this is the weight updation rule. So how to consider this case because we are considering the frequency the frequencies Fi. So suppose the feature vector I am giving one example the feature vector is suppose x1 that is mapped to Wi.

So that means only one time it is mapped. So corresponding to this the w_i^* if I consider this equation corresponding to this equation Wi will be equal to x1. Suppose the next feature vector the next feature vector is suppose x2 that is also mapped to Wi. So how many times two times. So if I consider that information two times in the above equation.

So $w_i^* = \frac{x1+x2}{2}$. So two times the feature vector is mapped to Wi. And suppose x3 is not mapped to Wi it is not mapped and we are considering like this. And suppose now x20 that feature vector is now mapped to Wi. So how many times three times because between x2 and x20 there is no mapping. So now three times it is mapped to Wi. So corresponding to this your $w_i^* = \frac{x1+x2+x20}{3}$ only three times it is mapped so divided by three with the help of this equation.

So equation already I have shown with the help of this equation I can determine this one. So that means we are determining the centroid. So this is very similar to the k-means clustering. So we can determine the centroid of the cluster corresponding to the input feature vector x1 x2 like this we can determine the centroid of the cluster. So this is called the frequency sensitive competitive learning. In case of the simple competitive learning the problem already I have explained that is the underutilization problem.

So only the winning neuron is updated and we are not considering the losing neurons. So to consider this problem we are considering another important network that is the Kohonen neural network. So let us move to the next slide. So we are considering the network is the Kohonen neural network that is nothing but the SOM the self organizing map. So the problem of the competitive learning is the underutilization problem.

So this problem is considered or this problem is addressed in case of the cohanan neural network. So corresponding to the winner we have to consider the weight updation rule. So how to determine the winner that already I have explained. So based on the minimum distance we can determine the winner and this is the weight updation rule. So this is the new resistor that is the new width the old width and the parameter $\in$ now we are

considering it is suppose $\in_{k,0}(x - wi)$ this is for the winner and for other neurons the weight updation rule is $w_j^* = wj + \in_{k,D}(x - wj)$. So for other neurons for winners this is the weight updation rule for other neurons this is the weight updation rule.

So here you can see the epsilon we are considering this parameter actually it is a function of two parameters one is k, k is nothing but the frequency. So that already I have defined in case of the frequency sensitive network k is the frequency Fi and we are considering the distance D is the distance between that is the Euclidean distance between Wi and Wj between these two weights Wi and Wj. So in the first expression here we are considering the distance 0 this distance is 0 because the distance between Wi and Wi is equal to 0. So distance between Wi and Wi is equal to 0 so that is why we are considering 0.

So in this case the distance is considered that is the neighborhood is considered. So neighborhood is considered around the winner and we are considering the Dmax that is the maximum distance we are defining maximum distance we are defining. So in this case what is the principle so suppose we are considering a particular neighborhood so this neighborhood we are considering and suppose this is the winning neuron the winning neuron. The winning neuron is Wi and we are considering other neurons suppose Wj so Wk suppose Wm so these neurons we are considering these weights Wi, Wk, Wm so these weights we are considering. So a particular neighborhood is considered around the winner so this Dmax is defined that is the distance is defined so that is the Dmax is defined. So this epsilon already I told you so it has two parameters one is k and another one is the distance here we are considering the distance is Dmax so Dmax is considered.

So in this case you can see the weight Wi that is updated like this so that is for the winner and we are considering the neighborhood and that is defined by Dmax so that means we have to update Wk, Wm because it is near to Wi and it is within the Dmax so that means we are updating Wk we are updating Wm so these weights I am updating because it is close to Wi that is the winning neuron. So now this epsilon it depends on k and Dmax so that is one is the frequency another one is the maximum distance this distance we are considering to consider the neighborhood around the weight vector Wi. So in this case the all the neighborhood weight vectors are updated along with the winners so winner takes all so this principle is called the winner takes all because first I have to determine the winners and after this around the winners we are also considering other neurons. So that means weight vector Wi is considered and all other weight vectors Wk, Wm within this particular neighborhood these are also considered for updation. So for updation we are considering all the neurons around the winning neuron so that is why it is called the winner takes all this is called the winner takes all.

So this is the concept of the Kohonen neural network so one problem in this case is in case of the competitive learning the problem was the underutilization so this problem is addressed in case of the Kohonen neural network but still there is one problem the problem I can show pictorially. So suppose my winning neuron is Wi that is a weight vector Wi corresponding to the winning neuron and suppose this is Wk this is Wj and suppose this is the feature vector x. So we are computing the distance from the weight vector so in this case if I consider this distance between Wi and Wj and also we are considering the distance between Wi and the Wk so the distance d1 and distance d2 so from the distance what it is updated Wj is updated first wj is updated first and then wk because wj is close to wi as compared to wk. So distance is measured from the winning node so wi is the winning node that is the winning weight vector and distance is measured from wi but actually the distance should be measured from the feature vector x. So if I consider this distance suppose this distance is suppose dj and this distance between x and wk that is suppose dk.

So in this case you can see dk is small than dj so if I consider the distance from x then what I have to consider wk should be updated first. So first I have to consider Wk and after this I have to consider Wj second so first I have to update Wk and after this I have to update Wj. So that means in case of the Kohonen neural network we are considering the distance from the winning neuron that is from Wi we are not considering the distance from the feature vector x. So that is the problem of the Kohonen neural network and that problem is eliminated in another network that is called the fuzzy Kohonen neural network. So that concept I am not going to explain that is the advanced concept so the problem this problem is eliminated in case of the fuzzy Kohonen neural network.

So this is the summary of the Kohonen neural network so what is the summary of the Kohonen neural network? So first we have to determine the winning neuron that is the corresponding weight vector we have to consider and this is the weight updation rule and Wj is nothing but the other neurons these are not the winners. So that means the winner is updated and all the losers are also updated and that means the topological neighborhood are also updated that means the weight vectors within this particular d max they are also updated along with the winning neurons. The learning rate monotonically decreases with increasing topological distances. So this topological distance already I have explained so this is the winning neuron Wi and this is the Dmax. So if I consider this weight vector and this weight vector so this learning rate monotonically decreases with the increasing of the topological distance.

So that means this learning rate corresponding to this and corresponding to this it depends on the topological distance. So the learning rate also decreases as training progresses and in case of the Kohonen neural network already I have explained the

concept is the winner takes all. So this concept I am explaining again so in case of the SOM or the Kohonen neural network I have explained that we are doing the mapping from the input space into a two dimensional array. So here I am considering the two dimensional array and we are showing the mapping from the input vector into a two dimensional array and we are determining the winning neuron and also we are considering the neighborhood neurons. So winning weight vector is updated as per the weight updation rule and all the neighborhood also they are also updated. So in case of the Kohonen neural networks we have considered that winner takes all and the lateral connections we are considering to develop a competition between the neurons of the network.

So we are employing some lateral connections to develop a competition between the neurons of the network. So that is also one important point the neurons having the largest activation level among all the output layer neurons is considered as the winner. So winner we have determined and we have to update the winners as per the weight updation rule. So this lateral connection concept I am explaining in my next slide this lateral connection is important to consider a competition between the neurons of the network. So here I have shown one function this function is called the Mexican head function.

So what is the objective of this function? The winning neuron is the only neuron which gives an output signal. The activity of all other network neurons is suppressed in this competition and we are considering the lateral feedback connections to produce excitatory or inhibitory effects depending on the distance from the winning neuron. So suppose we are considering this distance so all the neurons within this particular distance they are updated based on the distance and based on the frequency. But that is actually called the excitatory effect and if I consider that these neurons or that these neurons these are not updated. This is called the inhibitory effect. So we are considering the neurons around the winning neurons within a particular neighborhood and corresponding to these neurons we are considering the excitatory effect that means these neurons are also updated.

So that means these weight vectors are also updated in a particular neighborhood. So it depends on the epsilon. So epsilon is a function of two parameters one is the frequency and another one is the Dmax and corresponding to other neurons beyond Dmax we are not updating and that is called the inhibitory effect. So we are considering these two effects excitatory effect and the inhibitory effect and for this we are considering the Mexican hat function. So this is the concept of the SOM that is the self organizing map and that is the Kohonen neural network or the self organizing map. In this class I briefly explained the concept of supervised and unsupervised artificial neural networks.

In the supervised artificial neural networks I briefly explained the concept of multi-layer perceptron. After this I discussed the concept of back propagation training that is the back propagation learning and finally I discussed the concept of unsupervised clustering techniques. So for unsupervised clustering techniques I introduced the concept of competitive learning and the Kohonen neural network. In the competitive learning or in case of the Kohonen neural networks I have to update the winners. So I have to first find the winners and after this I have to update the winners.

In case of the competitive learning the problem is the under-utilization. So that problem can be addressed by Kohonen neural network. In the Kohonen neural networks I have to update both losing neurons and the winning neurons and finally I discussed the concept of fuzzy Kohonen neural network. So let me stop here today. Thank you.