Computer Vision and Image Processing - Fundamentals and Applications.

Professor Doctor M.K Bhuyan
Department of Electronics and Electrical Engineering
Indian Institute of Technology, Guwahati

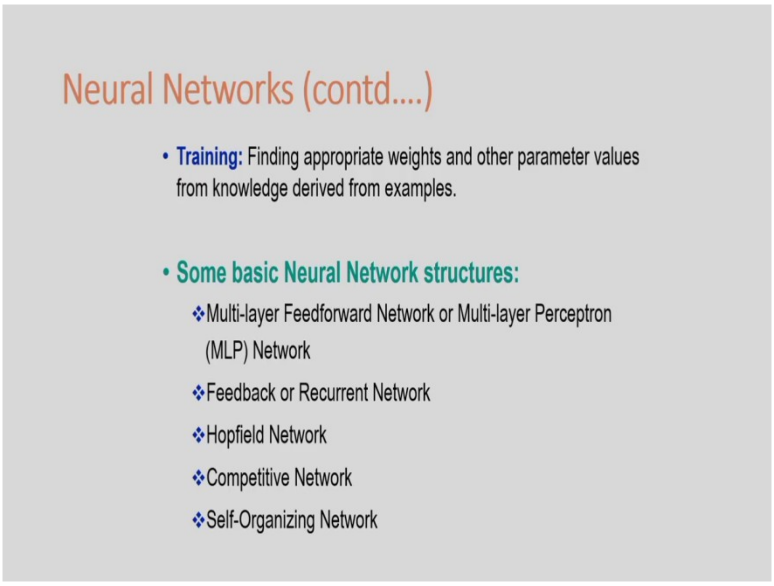
Lecture - 36

Artificial Neural Network for Pattern Classification

Welcome to NPTEL MOOCS course on Computer Vision and Image Processing Fundamentals and Applications. In my last class I discussed the basic concept of artificial neural network. Also, I highlighted the concept of supervised learning and unsupervised learning. And also, I mentioned the importance of hidden layer in an artificial neural network. For this I have given 1 example, implementation of AND and OR logic and also XOR logic that implementation I have shown.

For AND an OR I do not need any hidden layers, but if I consider the XOR logic in this case, I need 1 hidden layer between input layer and the output layer. Also, I discussed the concept of nearest neighbor classifier. Today I am going to discuss about different types of artificial neural networks. And also, I will discuss the concept of supervised learning and unsupervised learning. Let us see what are the different types of artificial neural networks.

(Refer Slide Time: 1:53)

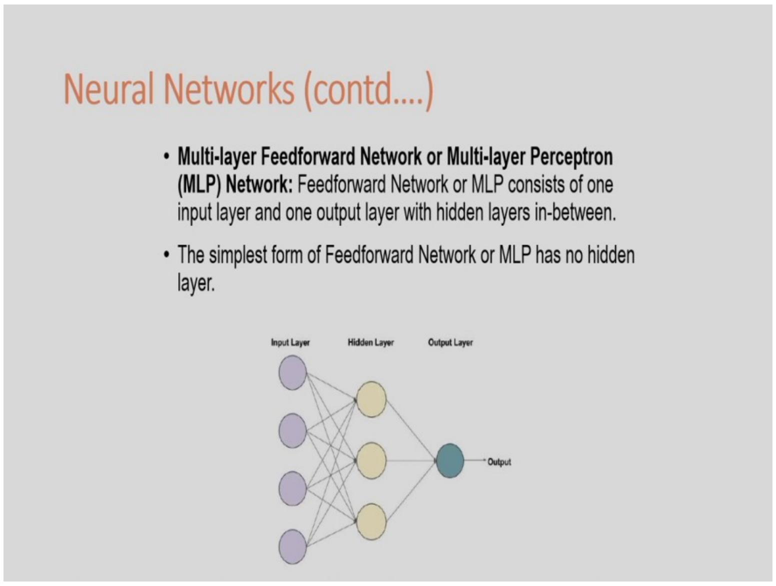


Neural Networks (contd....)

- **Training:** Finding appropriate weights and other parameter values from knowledge derived from examples.
- **Some basic Neural Network structures:**
  - ❖Multi-layer Feedforward Network or Multi-layer Perceptron (MLP) Network
  - ❖Feedback or Recurrent Network
  - ❖Hopfield Network
  - ❖Competitive Network
  - ❖Self-Organizing Network

So, this training concept I will explain later on that is about the supervised learning and the training is mainly to find appropriate weights and that knowledge of the artificial neural network is stored in the form of weights. That concept I have explained in my last class and some basic neural network structures I can mention, one is MLP that is multi layer feed forward network also it is called a multi layer perception, that is MLP. So, in case of MLP I have the input layer and also, I have the output layer in between I may have some hidden layers. After this the next one is the feedback or the recurrent network that I will explain, another one is the hopfield network, competitive network and the self organizing network.

(Refer Slide Time: 2:49)



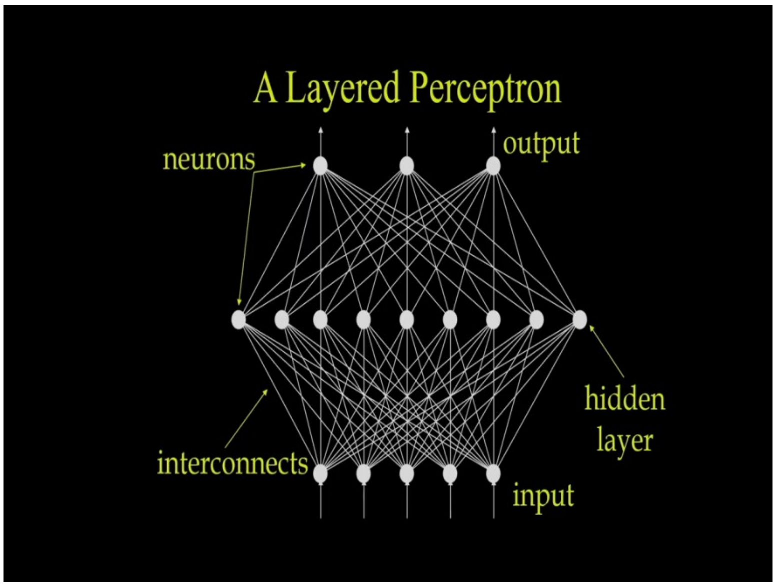
Neural Networks (contd....)

- **Multi-layer Feedforward Network or Multi-layer Perceptron (MLP) Network:** Feedforward Network or MLP consists of one input layer and one output layer with hidden layers in-between.
- The simplest form of Feedforward Network or MLP has no hidden layer.

Input Layer
Hidden Layer
Output Layer
Output

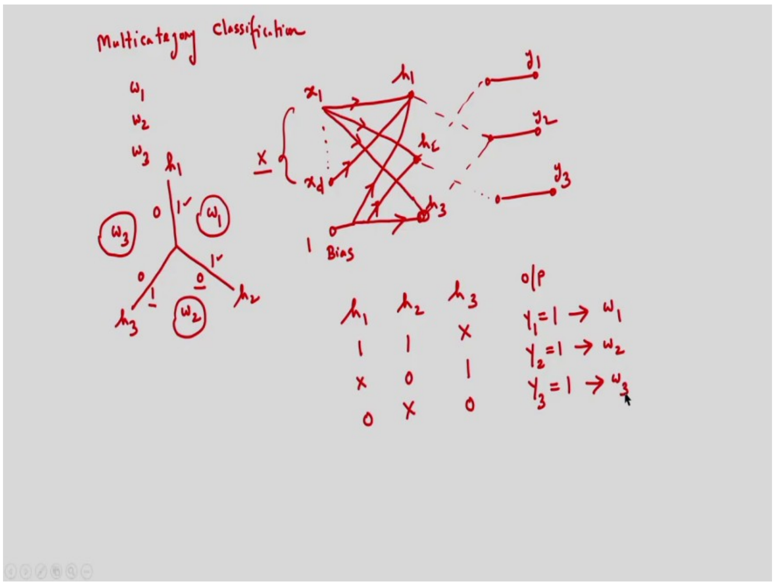
So, in case of the multi layer feed forward network that is the MLP, that multi layer perception. So, we have 1 input layer, 1 output layer and maybe the hidden layers in between. And in this case, if I consider the simplest form of feed forward network, then in this case, no hidden layer between input layer and the output layer. So, in this figure I have shown 1 MLP, that is the multi layer perception 1 input layer you can see 1 output layer and in between you can see 1 hidden, and layer that is the multi layer feed forward network.

(Refer Slide Time: 3:29)



And in this case also I have shown 1 feed forward network, you can see 1 is the input layer and in between the hidden layer is available and you can see the output layer. And you can see the interconnections between the input layer and the hidden layer and also the interconnections between hidden layer and the output layers. And also, you can see that neurons for the output layer, input layer and the hidden layers. So, this is one example of the feed forward artificial neural network.

(Refer Slide Time: 4:03)



Multicategory classification
ω_1
ω_2
ω_3
h_1
h_2
h_3
x_1
x_d
$\underline{x}$
Bias
y_1
y_2
y_3
o/p
$y_1 = 1 \rightarrow \omega_1$
$y_2 = 1 \rightarrow \omega_2$
$y_3 = 1 \rightarrow \omega_3$

So, I am giving one example of multi category classification. Suppose I am considering 3 classes w_1, w_2 , w_3, 3 classes I am considering and what is the importance of the hidden layer that I want to explain. So, suppose I am considering the decision boundaries like this and the hidden layers are h_1, h_2 and h_3. So, I can draw the neural network. So, suppose my input is x1 and I am considering the D dimensional feature vector. So, suppose this h_1, similarly, I have h_2 and also, I haveh_3, I am considering the hidden layers and maybe I may have one bias input. So bias input also I am considering.

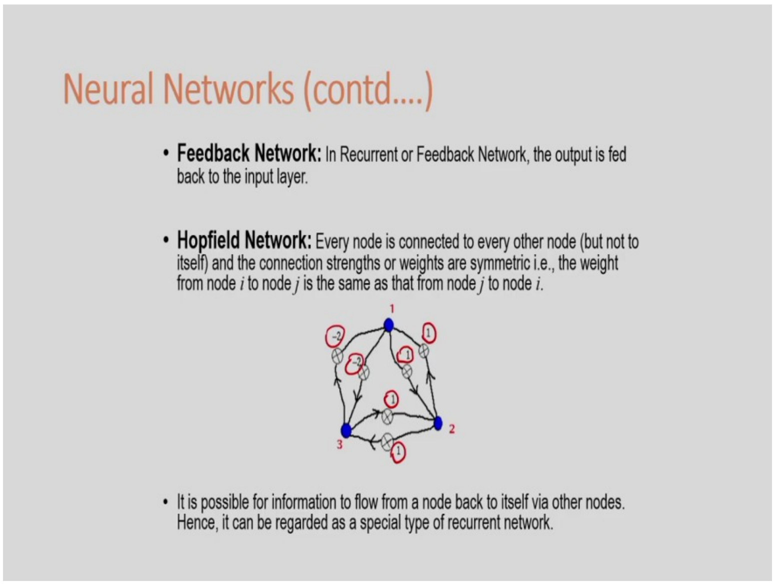
So, I am considering the bias also. And you can see the hidden layers, I am consideringh_1, h_2 , h_3 and after this I am not showing the interconnections, but output is suppose so y_1, y_2 and y_3 and between I have the interconnections, that interconnections I have not shown. So, h_1, h_2 , h_3, I am showing the hidden layers and input is the feature vector, the feature vector is x that is the D dimensional feature vector and I have the outputs output is y_1, y_2 and y_3.

So, suppose if I consider, suppose the hidden below h_1, h_2 and h_3. So, if I consider 1 and 1, h_1 is 1 and , h_2 is 1 that means, I am considering this one, $h_1$is 1 and h_2 is 1 corresponding to these suppose my class is suppose w_1. So, my output will be, my output will be y_1, y_1 should be equal to 1. The that corresponds to the class W 1 and H 3 do not care, any value, but h_1 should be 1 and , h_2 should be 1, $h_3$do not care. So, corresponding to this y_1 will be 1 and the corresponding class will be w_1.

Suppose if I consider h_1 is do not care, h_2 is suppose 0 and h_3 is suppose 1, that I can consider suppose y_2 is equal to 1 and that corresponds to the class w_2. So, that means this case in that case h_1 is do not care, but h_2 is 0, if you see it is 0 andh_3 is 1. So, corresponding to this the my class is w_2. So, this is my class, that class is w_2, that class is w_1 and suppose this class is w_3. So, corresponding to w_3, h_1 should be 0 andh_3 should be 0, h_2 do not care. So, corresponding to this y_3 will be equal to 1 and that corresponds to the class w_3.

So, you can see the importance of the hidden layer. So, based on the hidden layers, I am doing the classification, that is nothing but the multi category classification. So, in this example I have shown the importance of the hidden layer.

(Refer Slide Time: 8:37)



Neural Networks (contd....)

- **Feedback Network:** In Recurrent or Feedback Network, the output is fed back to the input layer.
- **Hopfield Network:** Every node is connected to every other node (but not to itself) and the connection strengths or weights are symmetric i.e., the weight from node *i* to node *j* is the same as that from node *j* to node *i*.
- It is possible for information to flow from a node back to itself via other nodes. Hence, it can be regarded as a special type of recurrent network.

Now, next one is our definition of feedback network, what is the feedback network. So, the feedback network, that is the output is fed back to the input layer. So, it is also called the recurrent neural network. Feedback networks can have signals travelling in both directions.

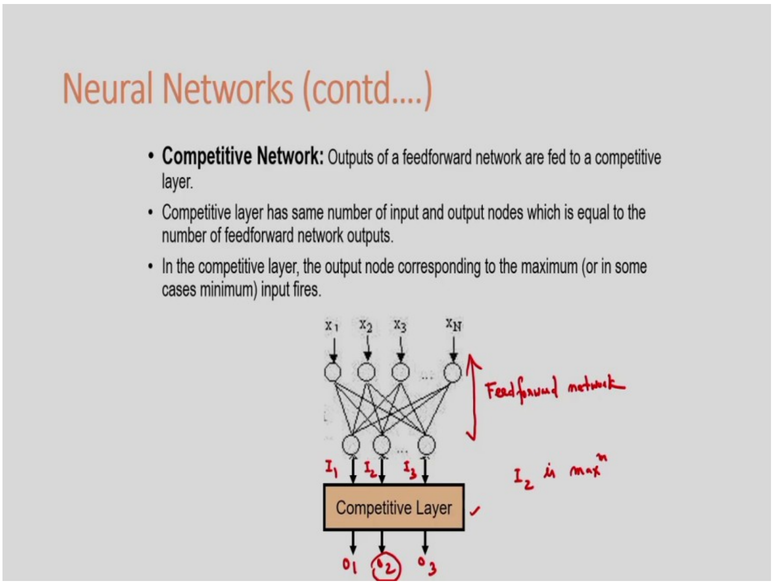
And this is implemented by introducing loop in the network. And the feedback networks are generally dynamic and their state changes continuously until they reach an equilibrium point.

And in this case, the recurrent network suppose if I consider. So, a recurrent network is one example of the feedback network. So, recurrent neural network uses feedback connections in single layer neural network architecture. So, that is the briefly the concept of the feedback network. That is the output is feedback to the input layer. And in case of a recurrent network, so recurrent network uses the feedback connections in single layer neural network architecture. After this the next one is the Hopfield network.

So, you can see the bigger the concept of the Hopfield network. Every node is connected to every other node, but not itself. So, here you can see in this figure, the node 1 is connected to node 3, node 1 is connected to node 2 like this and also the connection weights are symmetric. So, what is the meaning of this, the weight from node i to j is same as that from node j to i. So, in this figure if you see what is the weight from node 1 to 2, it is 1 and from 2 to 1 the weight is 1, that is the symmetric weights.

Similarly, if I consider the weights from the node 2 to 3, so, it is 1 and from 3 to 1 it is also 1 and similarly, if I consider the weights from node 1 to 3 that is minus 2 and from 3 to 1 it is minus 2. So, that is the symmetric weight. So, every node is connected to every other node and in this case, we are considering symmetric weights, that is the concept of the Hopfield network.

(Refer Slide Time: 11:10)



Neural Networks (contd....)

- **Competitive Network:** Outputs of a feedforward network are fed to a competitive layer.
- Competitive layer has same number of input and output nodes which is equal to the number of feedforward network outputs.
- In the competitive layer, the output node corresponding to the maximum (or in some cases minimum) input fires.

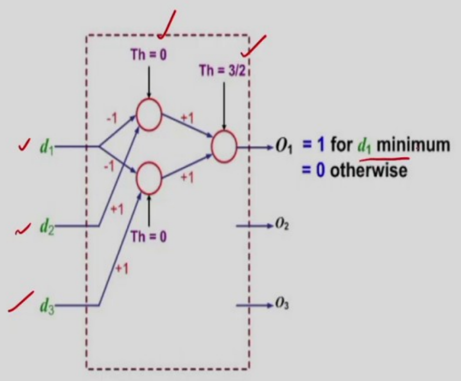
x_1
x_2
x_3
x_N
Feedforward network
I_1
I_2
I_3
I_2 is max^n
Competitive Layer
o_1
o_2
o_3

After this next one is the competitive networks. So, outputs of a feed forward networks are paid to a competitive layer. And in this case, you can see I am showing 1 feed forward network first part is the feedforward network, feedforward network and I have one competitive layer here you can see I have one competitive layer. So, in this case the competitive layer has the same number of inputs and the output nodes which is equal to the number of feed forward network outputs.

So, you can see this is suppose I_1, I_2, I_3. So, like this suppose this I_1, I_2, I_3, these are the inputs to the competitive layer, the competitive layer is nothing but a competitive. So, in the competitive layer the output node corresponding to the maximum input files. So, in suppose in this case, the output from the competitive layer is suppose O, O_2, O_3 these are the output from that competitive layers and suppose at a particular time suppose I_2 is maximum, I_2 is maximum out of I_1, I_2, I_3 , I_4 like this.

So, corresponding to this I will be getting the output, the output will be O_2 because the I_2 is maximum. So, I will be getting the output, output is O_2 and the competitive layer has the same number of input and the output nodes which is equal to the number of feedforward network outputs. And finally, what will be the output from the competitive layer, the output node corresponding to the maximum input files. So, that concept I have explained.

(Refer Slide Time: 12:59)



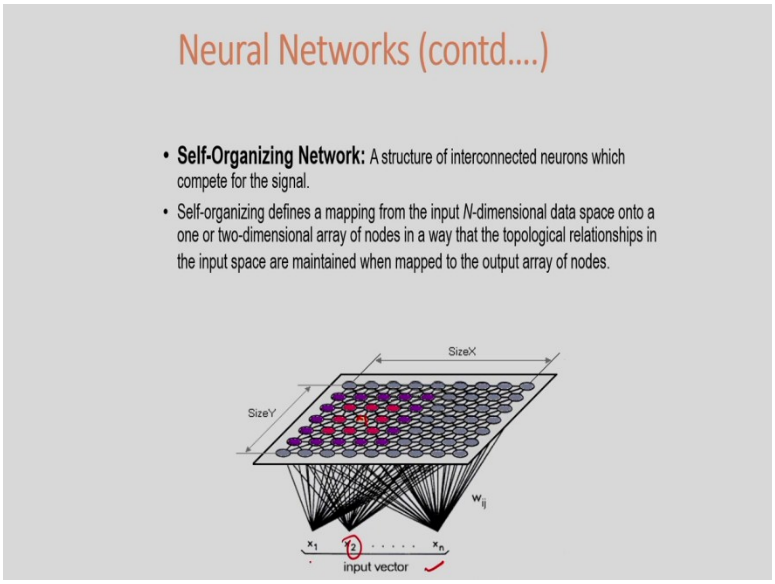
Competitive layer architecture

M.K. Bhuyan, Computer Vision and Image Processing – Fundamentals and Applications, CRC press, USA, 2019.

So, in this figure I am showing one architecture for the competitive layer. I am considering the inputs D_1, D_2, and D_3. So, 3 inputs I am considering and this structure can be used for determining minimum among 3 inputs using perceptions. So, you can see I want to determine the minimum between D_1, D_2, and D_3 at a particular time and this structure can be used for determining minimum among 3 inputs D_1, D_2, and D_3using perceptions. So, from this figure you can see, I want to determine the minimum of D_1, D_2, and D_3.

So, for this I am considering the first threshold, the threshold is equal to 0 and 2nd threshold I am considering, threshold is equal to 3 divided by 2. And you can see I am determining the minimum. So, suppose all 1 is equal to 1 that corresponds to D_1 is minimum, otherwise it will be 0.

(Refer Slide Time: 14:00)


Neural Networks (contd....)

- **Self-Organizing Network:** A structure of interconnected neurons which compete for the signal.
- Self-organizing defines a mapping from the input *N*-dimensional data space onto a one or two-dimensional array of nodes in a way that the topological relationships in the input space are maintained when mapped to the output array of nodes.

SizeX
SizeY
w_{ij}
x_1
x_2
x_n
input vector

And after this the next concept is the self organizing network. This is the interconnected neurons which compete for the signal and in this case you can see in the figure I have shown the input vector and the self organizing defines a mapping from the input n dimensional data space on to 1 or 2 dimensional array of nodes in a way that the topological relationships in the input space are maintained when mapped to the output array of nodes. So, meaning is you can see the input is the, suppose the n dimensional and data vector I am considering and I am doing the mapping, the mapping that is I am considering it is a 2-dimensional array of nodes.

And in this case, I am maintaining the topological relationship in the input space. So, here you can see the x_2 is close to x_1. So, that information I am maintaining after mapping. In this figure I am considering the n dimensional data space that is the input vector, I am mapped into a 2-dimensional array and also, I am maintaining that topological relationship in the input space. So, suppose x_2 is close to x_1, so, that information I am maintaining here.

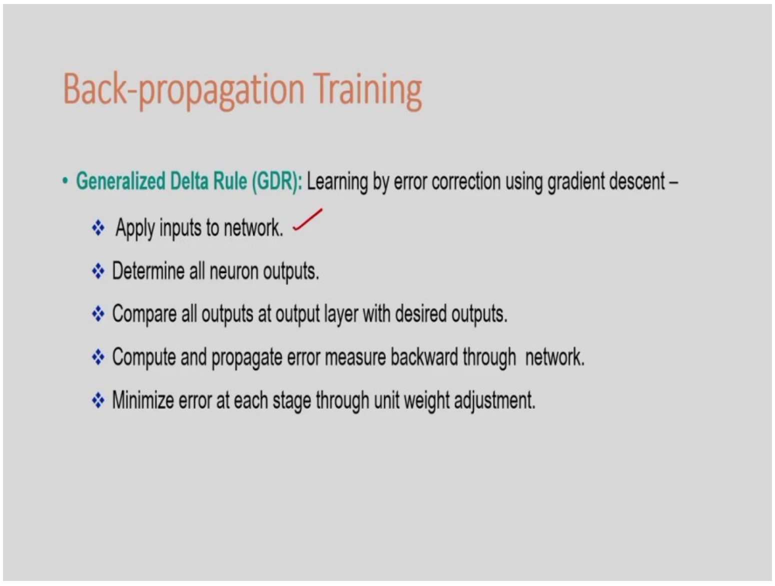
So, corresponding to suppose this is x_1 you can see, so, neighborhood will be x_2 like this. So, the topological relationships in the input space are maintained during the mapping. So, this is about the self organizing network. So, we will be discussing one important self organizing network, that is the kohonen self organizing network I will be explaining later on. So, this is very important, so, self organizing network that can be used for clustering of input data.

(Refer Slide Time: 15:55)

Supervised Learning

So, first I will be discussing the concept of the supervised learning and after this I will discuss the concept of the unsupervised learning in the artificial neural network. So, in case of the supervised learning, we know that desired output of the artificial neural network and also, we can determine the actual output from the neural network, the difference between these 2 is called the error. So, I have to minimize the error and for this, I am back propagating the error to the input and I can adjust the weights of the artificial neural network. The objective is to reduce the error, the error between the desired output and the actual output. So, that is the supervised learning.

(Refer Slide Time: 16:44)



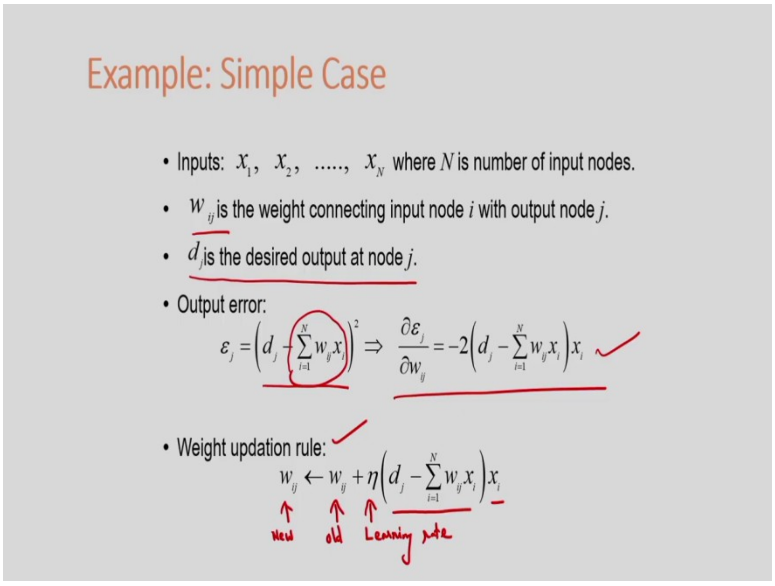
Back-propagation Training

- **Generalized Delta Rule (GDR):** Learning by error correction using gradient descent –
  - ❖ Apply inputs to network.
  - ❖ Determine all neuron outputs.
  - ❖ Compare all outputs at output layer with desired outputs.
  - ❖ Compute and propagate error measure backward through network.
  - ❖ Minimize error at each stage through unit weight adjustment.

So, the concept is the Generalized Delta Rule GDR. So, what I have to consider here, first I have to apply inputs to the network, after this determine all neuron outputs, I can determine the output from the neural network, after this compare all outputs at output layer with desired output. Since, we know the desired output corresponding to a particular training sample, that is why it is called the supervised learning. After this compute and the propagate error measure backward through the network.

So, objective is to minimize the error. So, for this I have to adjust the weights. So, minimize error at each stage through unit weight adjustment. So, I have to adjust the weights of the artificial neural network, so that the error will be minimum. That is the procedure of supervised learning and that is done generalized delta rule. And this is called the back propagation training, because the error is back propagated to the input.

(Refer Slide Time: 17:46)



Example: Simple Case

- Inputs: $x_1,\ x_2,\ \ldots..,\ x_N$ where N is number of input nodes.
- w_{ij} is the weight connecting input node i with output node j.
- d_j is the desired output at node j.
- Output error:

$$\varepsilon_j = \left(d_j - \sum_{i=1}^{N} w_{ij}x_i\right)^2 \Rightarrow \frac{\partial \varepsilon_j}{\partial w_{ij}} = -2\left(d_j - \sum_{i=1}^{N} w_{ij}x_i\right)x_i$$

- Weight updation rule:

$$\underset{\substack{\uparrow \\ \text{New}}}{w_{ij}} \leftarrow \underset{\substack{\uparrow \\ \text{old}}}{w_{ij}} + \underset{\substack{\uparrow \\ \text{Learning rate}}}{\eta}\left(d_j - \sum_{i=1}^{N} w_{ij}x_i\right)x_i$$

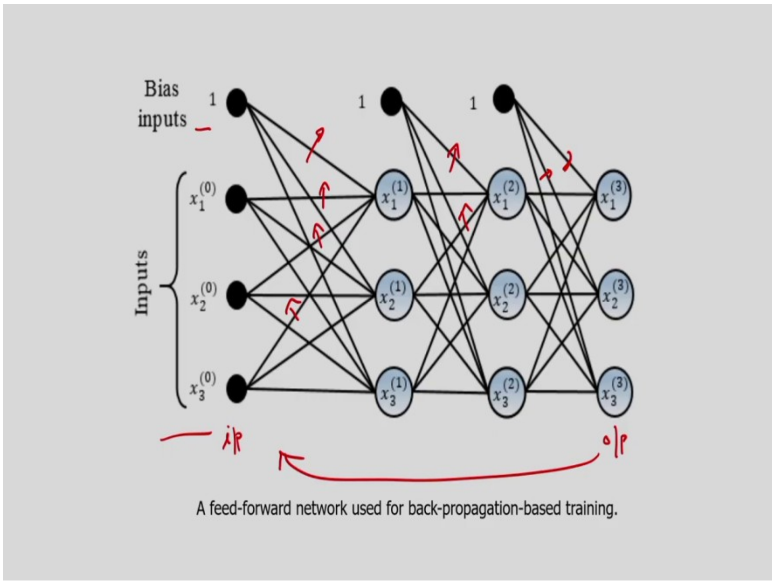
Then mathematically how can I explain this one. Suppose, an input I have the inputs $x_1 x_2 .. x_n$ and where n is the number of input nodes. So, I have the n number of input nodes and I am considering W_{ij}, that is the weights connecting input node i and the output node j. So, I am considering the weights of the artificial neural networks, the weights connecting input node i and the output node j. And also, we know the desired output at a particular node, then node is suppose j. So, we know the desired output. So, that information is available. So, that is why it is called a supervised learning.

And we can determine the output error, the output error is nothing but the desired output minus actual output. So, if you see this term, this term is nothing but I can compute the actual output from the artificial neural network. So, desired output minus actual output, that gives the error and I have to minimize the error and, in this case, I have to adjust the weights. So, that is why I am doing the differentiation of the error with respect to the weights, the weights are W_{ij}. So, I am doing the minimization of this.

So, after minimization you will be getting this one. Now, I am considering the weight updation rule. So, the rule is this one, weight updation rule. So, here you can see, this is the new weight I am considering, this is the old weight I am considering and here you can see this is nothing but the error, the error between the desired output and the actual output and xi is the input you can see. And one parameter I am considering that parameter is the eta, that controls the learning rate, that controls the learning rate. This parameter decides the rate of convergence, that parameter eta.

So, that means, suppose eta is small, the convergence will be slow, if eta high, the convergence will be fast. So, this is the weight updation rule for the back propagation training So, objective is to reduce the error and for this I have to adjust the weights.

(Refer Slide Time: 20:07)

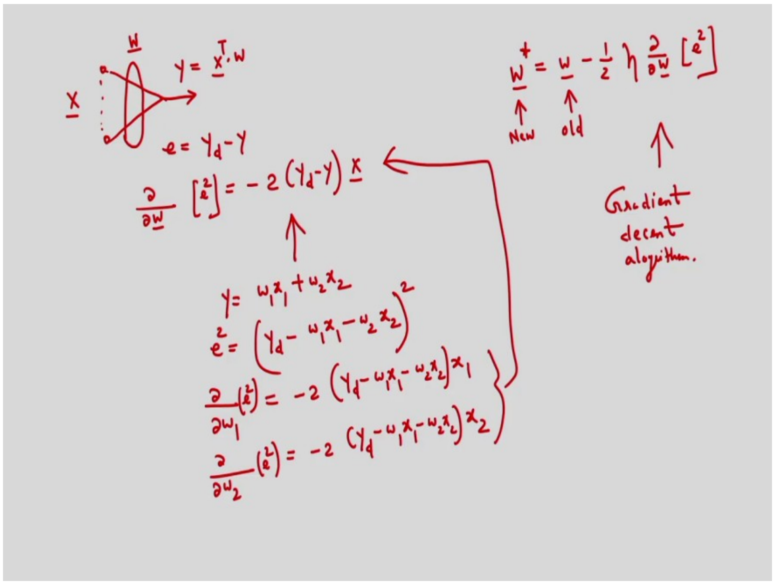


A feed-forward network used for back-propagation-based training.

So, here you can see I am showing one example of the backpropagation training, you can see I have the input layers and also, I am considering the bias inputs. And in between the input and the output layers, you can see this is the output layer and this is my input layer, we can have the hidden layers between the input layer and the output layers. And the concept is the same concept, we can have the desired output and we can determine the actual output in the network and we have to minimize the error. So, for this I have to adjust the weights.

So, weights are available here, all the weights I have to adjust, so that the error is minimum. So, the error is back propagated to the input, that is why it is called the backpropagation training and the error is back propagated to the input so that I can adjust the weights to minimize the error. So, this is the concept of the backpropagation training. So, this concept I can explain like this.

(Refer Slide Time: 21:12)



$\underline{x}$
$\underline{W}$
$y = \underline{x}^T \cdot \underline{W}$
$e = y_d - y$
$\frac{\partial}{\partial \underline{W}}[e^2] = -2(y_d - y)\underline{x}$
$y = w_1x_1 + w_2x_2$
$e^2 = (y_d - w_1x_1 - w_2x_2)^2$
$\frac{\partial}{\partial w_1}(e^2) = -2(y_d - w_1x_1 - w_2x_2)x_1$
$\frac{\partial}{\partial w_2}(e^2) = -2(y_d - w_1x_1 - w_2x_2)x_2$
$\underline{W}^+ = \underline{W} - \frac{1}{2}\eta\frac{\partial}{\partial \underline{W}}[e^2]$
New
old
Gradient decent algorithm.

Suppose I am considering a very simple network. So, suppose my input vector is suppose x and W is the weights of the network, the railway weight vector and output is y. So, y is nothing but x^TW, that is the dot product between the input vector and the weight. So, this is one simple network I am considering and, in this case, I can determine the error. The error is nothing but the desired output minus actual output, that is the error and after this, I have to minimize the error.

So, for this I have to differentiate this one,dw the error should be minimized. So, if I minimize this error, this will be $2(y_d - y)x$ I will be getting this one. So, how to get this expression, how to get these expressions, suppose y is equal to, suppose $y = W_1 x_1 + W_2 x_2$, I am considering 2 inputs suppose x_1 and x_2 and suppose I am considering weight W_1 and W_2. So, what will be my error now, the error square? The desired output I know and what is the actual output the actual output is $W_1 x_1$ plus it will be minus now, $W_2 x_2$.

So, error is desired output minus actual output and I am considering the squared error. I have to minimize the error, so this I have to take the differentiation with respect to the weight W 1. So, if I do the differentiation, then in this case it will be minus 2. So, $y_d - W_1 x_1 - W_2 x_2$and it will be x_1. Similarly, if I take the differentiation with respect to $W2$, the error should be minimized. So, in this case it will be -2 ($y_d - W_1 x_1 - W_2 x_2$) x_2 and this will be x_2.

So, from this you can see I will be getting this one. After this I have to adjust the weights. So, suppose this is my new weight, W star is the new weight, that is the weight vector, that is the old weight minus half eta I am considering. So, this I am considering 1 by 2. So, this is the new weight, this is the old weight and eta controls the learning rate, already I have explained, that is it controls the rate of convergence. If eta is small, the convergence will be slow. And if the eta is high that convergence will be fast.

And this is nothing but this what I am considering this is this algorithm is also called the gradient descent, gradient descent algorithm. Gradient Decent algorithm, because in this case I am minimizing the error, so for this I am determining the gradient. So, that is why it is called the gradient descent algorithms. So, you can see how to adjust the weights, how to get the new weight from the old weight. So, I have to minimize the error. So, this is the concept of the supervised learning and this is nothing but a back propagation algorithm.

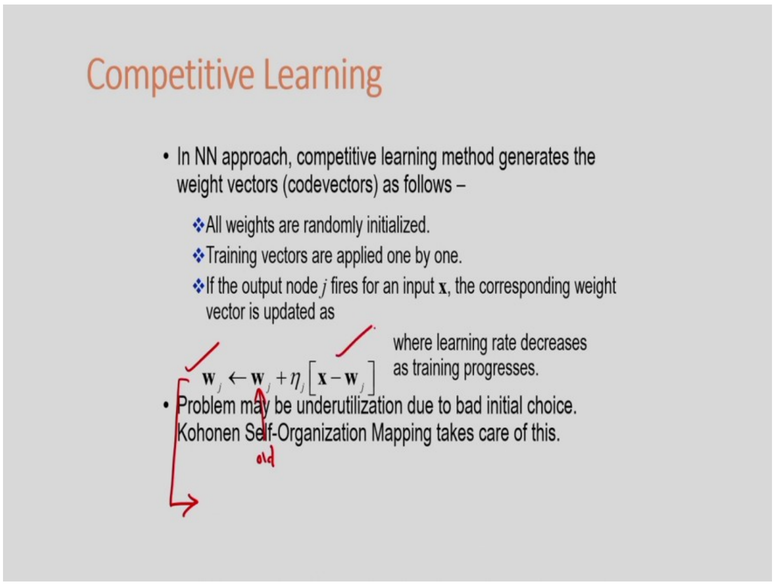
(Refer Slide Time: 25:32)

Unsupervised Learning

Next learning is the unsupervised learning. So, unsupervised learning is used for grouping of input data, that is nothing but the clustering, the clustering of input data. This the unsupervised learning process also term is the self organizing map. And in this case, in case of the unsupervised networks, in general, we do not use external input to adjust the weights. So, unsupervised learning, itself organizes data inputted to the network.

Unsupervised network look for some regularities or similarities of the input data and make adaptations according to the function of the network. And one important point is supervised learning is performed offline, while unsupervised learning is performed online. Now, I will explain some of the unsupervised learning techniques.

(Refer Slide Time: 26:31)



Competitive Learning

- In NN approach, competitive learning method generates the weight vectors (codevectors) as follows –
  - All weights are randomly initialized.
  - Training vectors are applied one by one.
  - If the output node j fires for an input $\mathbf{x}$, the corresponding weight vector is updated as

$$\mathbf{w}_j \leftarrow \mathbf{w}_j + \eta_j \left[\mathbf{x} - \mathbf{w}_j\right]$$

where learning rate decreases as training progresses.

old

- Problem may be underutilization due to bad initial choice. Kohonen Self-Organization Mapping takes care of this.

One is the competitive learning. Competitive networks cluster, encode and classify input data stream. That is, the objective is to classify an input pattern into one of the M number of classes. The network is learned to form classes of examplers, sample patterns, according to some similarities of these patterns. So, I have to find the similarities of the patterns and based on this, we have to form classes. Patterns in a cluster should have similar features. So, based on the similarity, we can group the patterns.

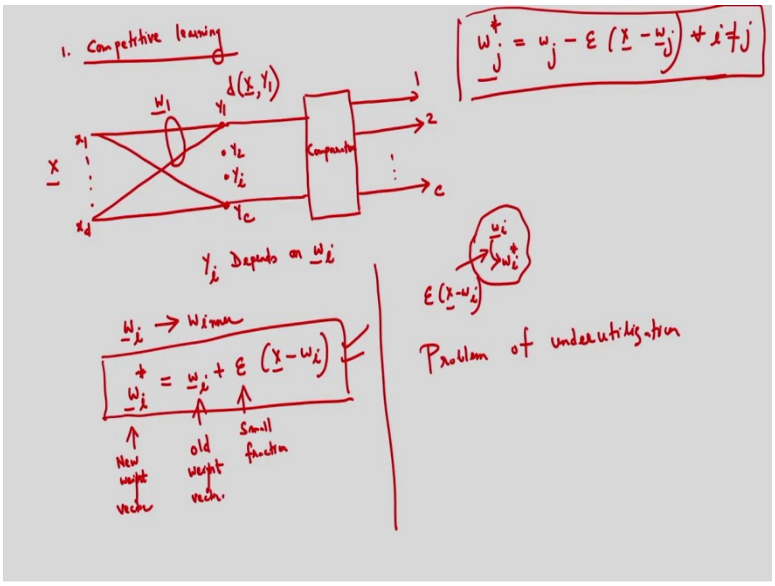
Now, in this case, the first one is you can see the competitive learning method generates the weight vector as follows. So, what is the method here you can see, all the weights are randomly initialized and training vectors are applied one by one. And if the output node j fires for an input x, the corresponding weight vector is updated, that means I have to determine the winner. And in this case, you can see the rule, the winner is updated and that is W_j is the new weight. And you can see that this is the old, this is the old weight.

And in this case, the winner is updated this concept I will explain, so what is the what is the competitive learning. So, the first the winner is determined by computing the distance between the weight vectors of the connecting neurons and the input vector. The winner is that neuron for which the weight vector has the smallest distance to the input vector. The square of the minimum equilibrium distance is used to select the winner.

So first, I have to select the winner. And subsequently, that weight vector W_i of the winning neuron is moved towards the input vector x. And this, you can see, the winner is updated by using this updation rule that I am going to explain now. So, first concept is I have to determine the winner and how to determine the winner. That is by computing the distance between the weight vectors of the connecting neurons and the input vector.

And in this case, I can consider the square of the minimum Euclidean distance. And after this after determining the winner what I have to do, the weight vector W i of the winning neuron is moved towards the input vector x. And is the output node i fires for an input vector x, the corresponding weight vector is updated. The updation is like this.

(Refer Slide Time: 29:29)



So, I am considering one network suppose, x is the input feature vector and this is a D dimensional feature vector. So, here you can see I am considering one, the competitive learning technique that is a competitive network, I am considering x is the input feature vector and this is a D dimensional feature vector and I am considering c number of classes. So, y 1, y 2, y c these are the cluster centers I am considering corresponding to c number of clusters y 1 y 2 y c, these are the cluster centers.

And corresponding to the first cluster center y 1 you can see, I am considering the weight vector, the weight vector W_1. So, in this case and this yi actually, it depends on Wi. That means the output depends on the weight vector, the weight vector is W_i. So, any output, that is the output the center of the cluster center depends on the weight vector W_i. And for this, for the competitive learning I have to determine the winner. So, for this I have to determine the distance between the input feature vector and the centroid.

So that I can consider the Euclidean distance. So, that means, I am determining the distance between the input vector x and that centroid. So, I have to find a winner which one is the winner, the minimum distance corresponds to the winner. Similarly, I can determine the distance between x and y to x and y see like this I can determine all the distances based on that minimum distance I can determine the winner. After this, after determining the winner, what I have to consider, we have to update the weight vectors.

Suppose, the winner is W_i that is that weight vector is W_i. So, I have to update the winner. So, corresponding to y i, my winning weight vector is W_i and in this case, I have to update the weight vector, the weight vector is W_i that is the winner. So, updation will be something like this. So, W_i is the winner. So, W_i it is updated like this. So, this is the new weight vector, weight vector, this is the old weight vector. And I am considering one small fraction parameter.

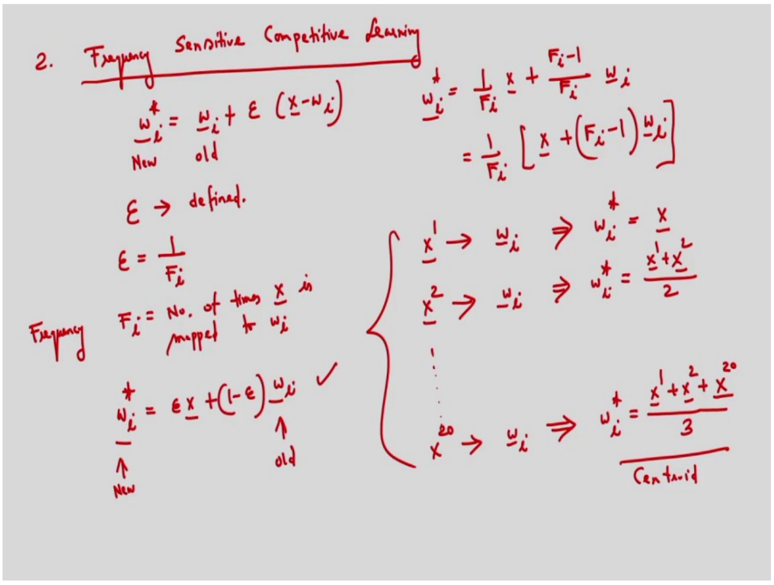
So, this is the weight updation rule I am considering. That means what I am doing since W_i is the winner, that means using W_i nearer to x and what about that weight updation rule for the other neurons. For other neurons the weight updation rule will be Wj is not the winner. So, what is the weight updation rule? The weight updation rule is $\epsilon x - W_j$. So, i is not equal to j for is not equal to j. So, this is for the loser. For the winner this is the weight updation rule for the loser, the weight updation rule is like this.

So, that means for the winner what I am doing, pushing Wi nearer to x and for the loser what I am doing pushing W_j away from x. So, this concept suppose if I consider and this is suppose W_i weight, so this is moved, W_i is moved to W_i 1 that is pushing W_i towards x. So that pushing by the amount, amount is x, pushing Wi towards x, and I am getting the new weight vector, the new weight is W_i. And in case of the loser what I am considering putting W_j, away from x.

So, that means the winner is updated, but the losers are not updated. So, our concept of the competitive learning is that first, I have to determine the winner based on the minimum distance, after this I have to update the winner, the winner is updated by using this formula. So, this is the winner is updated. That means what I am doing, that is pushing W_i nearer to x. And in case of the loser, this is the weight updation rule. That means what I am doing pushing W_j away from x.

So, what is the problem of this the competitive learning you can see here I am pushing Wi nearer to x, but what about the losers that means the pushing W_j away from x. So, this problem is called the problem of under utilization, this is the problem of, I can write this problem of under utilization, because the winner is updated, but the losers are not updated. So, like this, I am getting the clusters in case of the competitive learning.

(Refer Slide Time: 36:54)



2. Frequency Sensitive Competitive Learning

$$\underset{\text{New}}{\underline{w}_i^*} = \underset{\text{old}}{\underline{w}_i} + \varepsilon(\underline{x} - \underline{w}_i)$$

$\varepsilon \rightarrow$ defined.

$\varepsilon = \frac{1}{F_i}$

Frequency $F_i =$ No. of times $\underline{x}$ is mapped to $\underline{w}_i$

$$\underset{\uparrow \text{New}}{\underline{w}_i^*} = \varepsilon \underline{x} + (1-\varepsilon) \underset{\uparrow \text{old}}{\underline{w}_i} \checkmark$$

$$\underline{w}_i^* = \frac{1}{F_i}\underline{x} + \frac{F_i - 1}{F_i}\underline{w}_i$$

$$= \frac{1}{F_i}\left[\underline{x} + (F_i - 1)\underline{w}_i\right]$$

$\underline{x}^1 \rightarrow \underline{w}_i \Rightarrow \underline{w}_i^* = \underline{x}$

$\underline{x}^2 \rightarrow \underline{w}_i \Rightarrow \underline{w}_i^* = \frac{\underline{x}^1 + \underline{x}^2}{2}$

$\vdots$

$\underline{x}^{20} \rightarrow \underline{w}_i \Rightarrow \underline{w}_i^* = \underbrace{\frac{\underline{x}^1 + \underline{x}^2 + \underline{x}^{20}}{3}}_{\text{Centroid}}$

Next, I am considering another network and that is called the frequency sensitive, frequency sensitive competitive learning. So, in competitive learning, we define the updation rule is like this Wi is equal to Wi plus epsilon x minus Wi. So, this is the old weight, this is the old weight and this is the new weight. In case of the frequency synthetic quantitative learning, what I am considering this parameter the epsilon parameter that is defined, this epsilon parameter is defined.

So, epsilon is defined like this epsilon is equal to 1 by Fi. So, what is Fi, Fi is number of number of times x is mapped to Wi. So, Fi is number of times x is mapped to Wi, that means, the number of times Wi is the winner. So, frequency is defined and based on the frequency I am defining the epsilon, that small fraction I am considering epsilon is a small fraction. So, epsilon is equal to 1 by Fi. So, that means, I can write W i star is equal to epsilon x plus 1 minus epsilon Wi. So, this is the weight updation rule for frequency sensitive competitive learning.

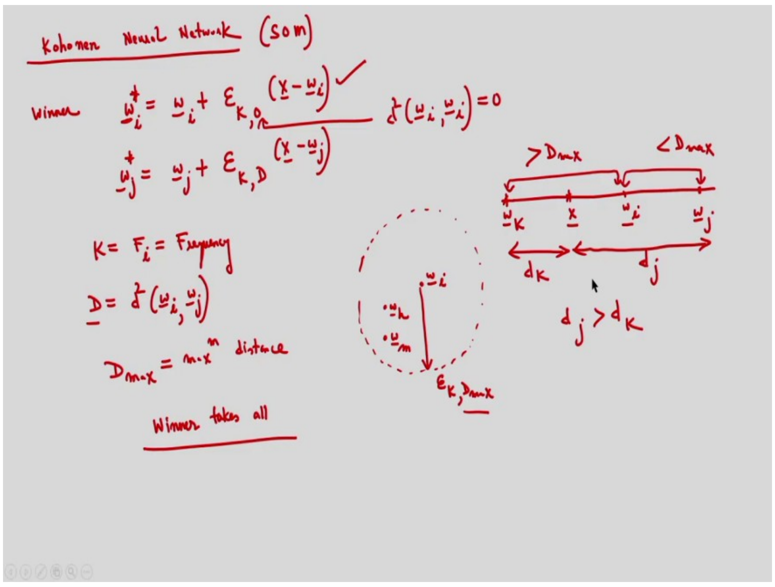
So, like this, this is my new weight, this is the old weight and I am considering this is the weight updation rule and in this case and this is nothing but the Wi is equal to 1 by Fi because epsilon is equal to 1 by Fi, Fi is the frequency x plus Fi minus 1 Fi Wi is equal to 1 by Fi x plus Fi minus 1 Wi. So, this is the weight updation rule. If I give one example suppose x 1 is mapped into the weight vector the weight vector Wi. So, that means I am considering number of times and Wi is the winner. So, that means the number of times the x is mapped into Wi.

And corresponding to this what will be my weight updation rule, the weight updation rule will be Wi is equal to x because only one time Wi is the winner. Suppose, next one is x 2, I am considering, next feature vector I am considering and in this case 2 times Wi is the winner. So, that means, what will be the new weight, the new weight will be x 1 plus x 2 divided by 2, that is my new weights. And after this suppose the next x 20 suppose I consider x 20, that is mapped into W i, in between there is no mapping suppose.

So, first mapping is x 1 is mapped into Wi, that means 1 time Wi is the winner. Next x 2 is mapped into Wi that means 2 times Wi is the winner. In between there is there is no mapping and after this x 20 is mapped into Wi that means, how many times Wi is the winner, the 3 times. So, that means the new weight will be x 1 divided by 3. So, like this I can determine the Wi star, that is the new weight vector I can determine.

And in this case, it is nothing but the centroid you can see it is the centroid I am determining. So, in this example what I am considering So, 3 times Wi is the winner, that means the frequency I am defining, frequency Fi. So, in this example 3 times it is mapped into Wi. So, that is why based on this I am determining the updated or the new weights I am determining and that is nothing but I am determining the centroid. So, this is the concept of the frequency sensitive competitive learning and that is one variant of the competitive learning.

(Refer Slide Time: 42:39)



Kohonen Neural Network (SOM)

Winner $\underline{w}_i^{*} = \underline{w}_i + \varepsilon_{K,0}(\underline{x} - \underline{w}_i)$ ✓ $d(\underline{w}_i, \underline{w}_i) = 0$

$\underline{w}_j^{*} = \underline{w}_j + \varepsilon_{K,D}(\underline{x} - \underline{w}_j)$

$K = F_i = \text{Frequency}$

$\underline{D} = d(\underline{w}_i, \underline{w}_j)$

$D_{max} = \max^m \text{ distance}$

Winner takes all

$\underline{w}_i$
$\underline{w}_h$
$\underline{w}_m$
$\varepsilon_{K,D_{max}}$
$> D_{max}$
$< D_{max}$
$\underline{w}_K$
$\underline{x}$
$\underline{w}_i$
$\underline{w}_j$
d_K
d_j
$d_j > d_K$

The next competitive learning is the Kohonen neural network, Kohonen neural network, that is also called a Self-Organizing Map, that is also called SOM. So, what is the concept of the Kohonen neural network? The weight updation rule will be something like this. So, Wi star is equal to Wi plus epsilon k 0 x Wi. This is for that winner, first I have to determine the winner based on the minimum distance. So, suppose Wi is the winner. So, for winner, this is the weight updation rule.

And for the loser that weight updation rule is W j plus epsilon k D. In case of the competitive learning the winner is updated, it is moving towards the feature vector x, but the losers are not updated, they are moving away from x, that is the problem and that problem already I have explained that is called the problem of under utilization. So, that problem is eliminated in kohonen neural network, that is the self organizing neural network. So, here you can see for the winner, this is the weight updation rule, Wi star is equal to Wi plus epsilon k 0 x minus Wi.

And Wj is equal to Wj plus epsilon k d. So, here you can see, now epsilon is a function of two parameters one is K and other one is D. So, what is k? k is nothing but the frequency and what is D, the D is the distance between Wi and Wj. So, D is the distance between Wi and Wj. So, in case of these if I consider the first one is in case of the winner, the updation rule is this and why I am considering it is 0, because the distance between Wi and Wi will be 0. So, that is the distance between the Wi and the Wi it will be 0. So, that is why I am considering it is 0.

But if I consider the distance between Wi and Wj, that distance is D. In case of the kohonen neural network, I have to define, the distance the maximum distance I have to define. So, this is the maximum details D max is the maximum distance and what actually I am doing here, you can see I am considering a neighborhood around a winner suppose the winner is Wi, winner is updated. And here you can see I am considering the distance, I am considering the epsilon, epsilon is a function of k n d.

So, I am considering the distance the D max I am considering. So, I am considering the neighborhood, neighborhood is suppose W h 1 weight vector is W h, another weight vector is suppose W k suppose or maybe suppose W m suppose. So, here you can see that winner is updated and also the weight vectors which are neighborhood of the weight vector Wi they are also updated. So, W h will be updated and W m will be updated. So, that means all the weight vectors near to that weight vector Wi will also be updated and here you can see the epsilon is a function of frequency and the distance.

And in this case, you can see, so up to D max, up to D max, that is the maximum distance we can update all that weight vectors and beyond this I am not updating. That means D max the learning vector will be 0, that means no updating. And the weight vectors away from Wi will be less updated. But after D max, there will be no updating. So, up to D max I can update all the neighborhood weight vectors.

So, winner I have to consider based on the minimum distance and unlike the competitive neural networks only the winner is updated but here winner is updated and all the other weights, that is the neighborhood of Wi will be updated but based on the frequency and based on the distance. So, I am repeating this, that is unlike competitive learning the weight associated with other nodes within his topological neighborhood are also updated along with the winner node.

And in case of the kohonen and neural network that is a self organizing map it is called topology preserving maps, because it assumes a topological structure among the cluster units. The self organizing defines a mapping from the input n dimensional data space onto a one- or two-dimensional array of nodes in a way that the topological relationship in the input space are maintained. So, a self organizing map is a structure of interconnected neurons, which compete for a signal. So that concept already explained.

So, that is why the self organizing neural networks are also called the topological preserving maps. It assumes a topological structure among the cluster units. And unlike competitive learning, the weights associated with other nodes within each topological neighborhood are also updated along with the winner node. So, that means in competitive learning, only the winner is updated, the losers are not updated.

But in case of the kohonen neural network or a self organizing map the winner is updated and also the neighborhoods are also updated. So, that is the concept and the learning rate monotonically decreases with increasing topological distance and the learning rate also decreases as training progresses. The winning neuron is considered as the output of the network and in this case the concept is winner takes all. Because first I have to determine the winner and after this the winner is updated and the neighborhood around the winner is also updated.

So, that is why it is called winner takes all, this is called the winner takes all. In the sum, nodes compete with each other using the strategy winner takes all. And I will be showing the some lateral connections later on. But this is the concept of the kohonen neural network. So, the problem of under utilization is eliminated, in case of the competitive learning the problem is the problem of under utilization. But in case of the kohonen neural network, I am considering the neighborhood. So, that is why the problem of under utilization is eliminated.

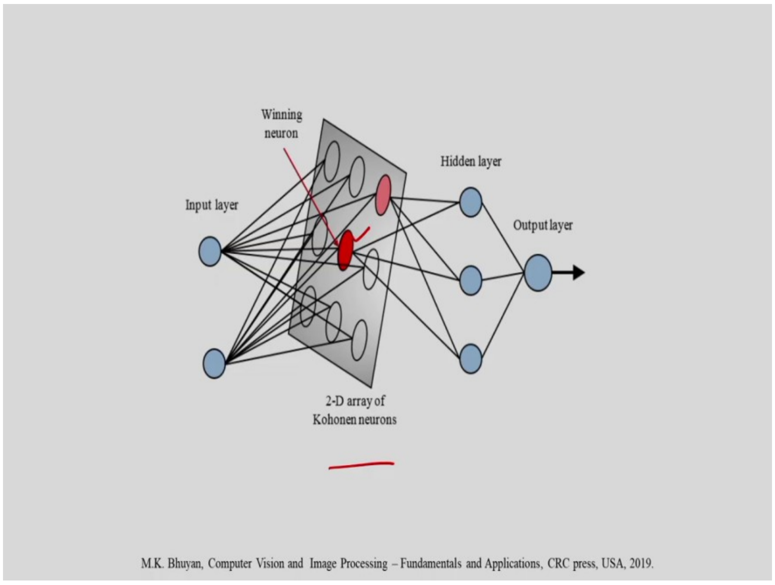
Now, one problem is here. So, suppose this is Wi, the winning node, the winning neuron and the corresponding weights. So, Wi is the winner suppose here and suppose the feature vector is x and suppose this is W k and it suppose Wj. So, here you can see the distance is computed from the winner, not from the feature vector. So, here you can see the distance between Wi and Wj, that is less than D max and suppose that this distance is greater than D max.

So, that means, that W j will be updated, but Wk is not updated because the distance is measured from the winner, the winner is Wi. That is why it is called a winner takes all, but the distance should be measured from x that is the feature vector. So, if I consider this distance, this distance is suppose Dj and suppose this distance this distance is suppose DK. So, in this case you can see the Dj is greater than DK.

So, if you consider this distance, the distance from the feature vector, then in this case the Wk should be updated first and after this the Wj should be updated. Because the distance from X to W k is smaller than the distance from X to W j. So, that is why the W k should be updated and after this we can update W j. But since in this case, we are determining the distance from the winner. So, that is why the W j is updated first and after this the Wk may not be updated, because it is greater than D max.

But the problem is here that we are not determining the distance from the feature vector, the feature vector is x, that is the problem of the kohonen neural network. So, this problem can be eliminated by considering another competitive network, that is called the fuzzy kohonen neural network.

(Refer Slide Time: 53:13)

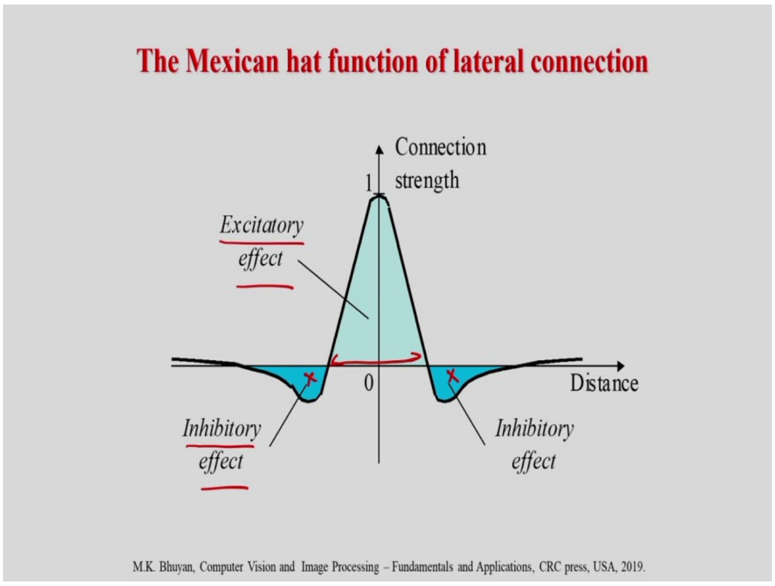


M.K. Bhuyan, Computer Vision and Image Processing – Fundamentals and Applications, CRC press, USA, 2019.

In this case, I am showing one structure of the self organizing map that is the kohonen neural network I am showing. You can see, we have the input layer, we have the hidden layer and we have the output layer. And also, you can see the mapping from the input layer to the 2d array. The input is the suppose n dimensional vector, and I am considering the 2d array and you can see I am considering the winning neuron. So, in the SOM, nodes compete with each other using the strategy, the strategy is winner takes all.

And I can consider some connections, these connections are called the lateral connections, the lateral connections are employed to develop a competition between the neurons of the network. The neurons having the largest activation level among all the output layer neurons is considered as the winner. The activity of all other network neurons is suppressed in the competition process. I am considering lateral feedback connections which are used to produce excitatory or the inhibitory effects depending on the distance from the winning neuron of the network. And this is implemented by using a Mexican head function. So, an in the next slide I can show what is the Mexican head function and that is used to produce excitatory or inhibitory effects.

(Refer Slide Time: 54:49)



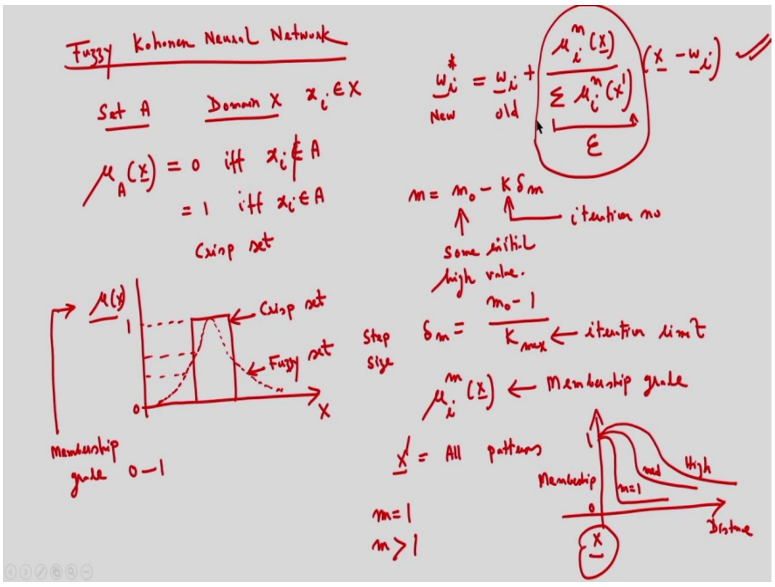
The Mexican hat function of lateral connection

M.K. Bhuyan, Computer Vision and Image Processing – Fundamentals and Applications, CRC press, USA, 2019.

So, let us see what is the Mexican hat function, here you can see. So, by considering this Mexican hat function, that we are considering the lateral connections, which can produce excitatory effect, you can see here the excitatory effect or we can produce the inhibitory effect. So, I am repeating this. So, this Mexican hat function describes the distribution of the weights between the neuron in the kohonen layer. So, because I have to consider the neighborhood weights, so, that is why, so, we are considering the neighborhood weights.

But after D max, I am not considering the weights. So, that is why I am considering the inhibitory effects. For the excitatory effects, we are considering the neighborhood weights around the winning neuron. The problem of the kohonen neural network is that the distance is computed from the winner, but distance should be computed from the feature vector. So, that is why to overcome this problem, I can consider another competitive network that is called a fuzzy kohonen neural network.

(Refer Slide Time: 55:56)



So, that concept I am going to explain now, so, that is called the fuzzy kohonen neural network. So, simply I want to first explain the concept of the fuzzy set and the normal set. So, suppose the set is suppose A and suppose I am considering the domain X. So, $x_i$is an element of $X$and domain I am considering is X is the domain and set I am considering A is the set. So, I am considering one function, that is the membership function I am considering.

μ_A(X) that is the membership function of the set A is equal to 0, if $x_i$is not an element of A is equal to 1 if $x_i$is an element of A. This is the case of the crisp set. So, for the crisp set it will be either 0 or 1. So, I can show the pictorially, so what is the meaning of this. So, suppose the membership grade I am considering x and you can see the crisp case is something like this. So, either it will be 1 or it will be 0. But in case of the fuzzy set you can see, I have the membership function something like this. This is a membership function.

So, this is the Crisp set our crisp set. And I can consider this is a fuzzy set. You should read the concept of fuzzy logic from a book on fuzzy logic. But briefly I am explaining the concept of the crisp set and also the fuzzy set. In case of the crisp set, you can see either I have the 0 value or 1 value. But in case of the fuzzy sets, based on the membership function, the membership grade and this is called a membership grade, membership grade I will be getting the value 0, 0.1, 0.2, 0.3, 0.4, 0.5, like this I will be getting, so all the values I will be getting based on the membership grade. This membership grade lies between 0 and 1.

So, this concept better you should read from a book on fuzzy logic. Briefly I am explaining the concept of the fuzzy set and the crisp set. Now what is the fuzzy Kohonen neural network because we have seen the problem of the kohonen neural network because the distance is measured from the winner. In case of the fuzzy Kohonen neural network, what I am considering the updation rule is something like this. The weight updation rule is $W_i = W_i + \mu_i(X)$.

So, this is the weight updation rule for the fuzzy kohonen neural network. So, this is the new weight you can see this is the new weight and this is the old weight, this is the old weight. And in place of epsilon, because in case of the competitive network it was epsilon, I am considering this one. So, what I am considering what is the membership grade I am considering here the $\mu_i(m)$. is the membership grade and, in this case, what is m, m is nothing but m not minus k delta m a I am considering.

So, some initial value I am considering some initial high-value I am considering and K nothing but the iteration number, iteration number I am considering and delta m is the step size. What is $\nabla M ? \nabla M$ is equal to $m_0 -$¿1 K max. So, it is iteration limit. So, maximum iteration I am considering, So, ∇M is the step size and in this case what I am considering the membership grade, this is the membership grade, membership grade I am considering.

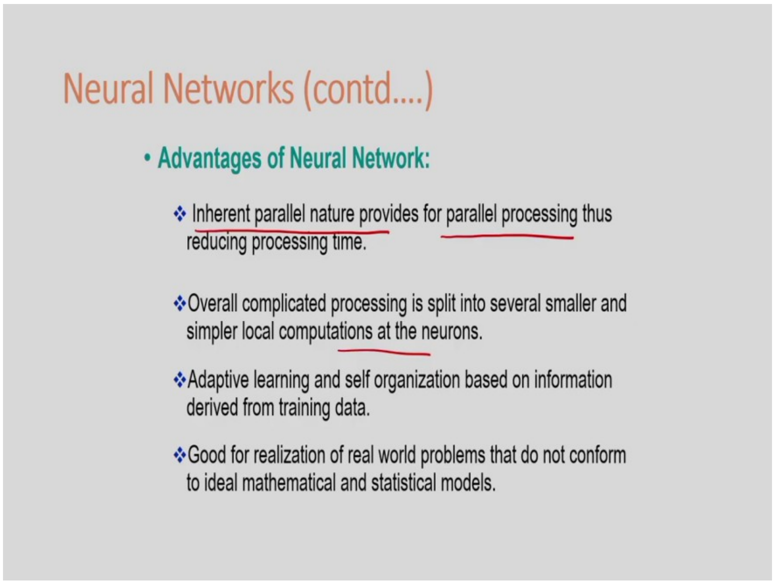
So, I am defining the weight updation rule, the weight of updation rule is like this. So, in place of epsilon I am writing this 1 mu i m x divided by summation mu i m x dash. So, what is x dash actually we are considering all the patterns, x dash means all the patterns I am considering all the patterns I am considering. And in this case, what is the importance of this m you can see. So, I am just plotting a membership, I am plotting the membership against the distance from the feature vector, feature vector is x.

So, corresponding to m is equal to 1, I have this one corresponding to the m is medium suppose, medium is this one, i is suppose this. So, corresponding to m is equal to 1, I am not considering any neighborhood, but if I consider that means corresponding to m is equal to 0, corresponding to m is equal to 1 I am not considering any neighborhood. But if I consider m is greater than 1, that means I am considering that neighborhood. So, suppose m is very high, that means, I am considering more and more neighborhood.

So, in this case instead of considering the D max that I have defined in the kohonen neural network, here I am considering m and if the m is greater than 1, that means, if I consider a very high value, that means, I am considering more and more neighborhood. And if I consider m is equal to 1, that means I am not considering any neighborhood. And in this case, you can see the distance I am computing from x. Distance is not computed from the winner and corresponding to m is equal to 1 membership will be 0, that means in this case, I am not considering the neighborhood.

But if I consider the m is greater than 1, then that means based on this membership function you can see here, I am considering more and more neighborhood. So, that I can put in this expression, this membership grade is available here. So, this is the concept of the fuzzy kohonen neural network.

(Refer Slide Time: 64:31)



Neural Networks (contd....)

- Advantages of Neural Network:
  - ❖ Inherent parallel nature provides for parallel processing thus reducing processing time.
  - ❖ Overall complicated processing is split into several smaller and simpler local computations at the neurons.
  - ❖ Adaptive learning and self organization based on information derived from training data.
  - ❖ Good for realization of real world problems that do not conform to ideal mathematical and statistical models.

And finally, I want to summarize. So, what are the advantages of the artificial neural networks.Once important thing is the parallel processing. So, inherent parallel nature provides parallel processing. And the second advantage is that the overall complicated processing is split into several smaller and simpler local competitions at the neurons. So, that is the second advantage. The first advantage is the parallel processing, after this we are considering the adaptive learning and the self organization based on information derived from the training data.

So, that is the concept of the adaptive learning and the self organization already I have explained, that is by using the self organization we can do the grouping. So, suppose the input feature vector is available and based on the similarity, we can do the grouping. Another advantage is that good for the realization of the real old problems that do not conform to ideal mathematical and statistical model. So, suppose one problem is there, we cannot develop mathematical or statistical models.

So, for this we can apply artificial neural networks. So, these are the advantage of the artificial neural networks. So, in this class, I discussed the concept of supervised learning and unsupervised learning. For supervised learning, I considered the back propagation neural network, the back propagation training. For this what we have considered, we know the desired output and also, we can compute the actual output. So, from this we can determine the error.

And the error is back propagated to the input, so that we can adjust the weights of the artificial neural network, that is the concept of the backpropagation learning. After this I considered some unsupervised learning techniques. The first one is the competitive learning. So, for this I have to determine the winner based on the distance between the feature vector and the centroid. And after this, I discussed one variant of the competitive learning, that is the frequency sensitive competitive learning.

And finally, I discussed the kohonen neural network, that is the self organizing map. For this we have to again determine the winning neurons. And also, we have to consider the updation of the winners and also the neighborhood and the concept is winner takes all. And finally, I discussed the concept of the fuzzy kohonen neural network. So, let me stop here today. Thank you.