

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

Lecture - 28
Some Examples of Neural Networks (Contd.)

(Refer Slide Time: 00:15)

Self-Organizing Map (SOM)/ Kohonen Network

- Proposed by T. Kohonen before 1995
- It can be used as visualization technique or dimensionality reduction technique (topology preserving tool)
- It can be used as a clustering algorithm also
- It works based on unsupervised and competitive learning

Handwritten notes: D.R.T, Distance preserving, Topology preserving, Div

Diagram: A diagram of a Self-Organizing Map showing an input layer, a competition layer with a winning neuron, and connections through synaptic weights. A handwritten 'D.R.T' is next to the diagram.

Self-Organizing Map

Now, we are going to discuss the working principle of another Neural Network, which is known as the Self-Organizing Map and in short, this is known as SOM. Now, this network was proposed by Kohonen around 1994/95 and according to his name, this network is also known as Kohonen network. Now, here we will see that in this particular network actually, we use the principle of unsupervised learning. So, the concept of that unsupervised learning, I am going to discuss in details.

Now, before that let us mention that this self-organizing map can be used as a visualization technique or a dimensionality reduction technique and this is actually a topology preserving tool. Now, let me discuss a little bit, what do you mean by the visualization technique or dimensionality reduction technique and what do you mean by this topology preserving tool? Now, supposing that say I have got a data in the higher dimension, say in 10 dimensions, 20 dimensions, something like this.

Now, this higher dimensional data can be represented something like this. So, this is actually the representation of the higher dimensional data, a large number of data and supposing that, so this data set is having say L dimensions, say 10 dimensions.

Now, this data we, human-beings, we cannot visualize the reason is: we can visualize only up to three dimensions. Now, if the data are in more than 3 dimensions, say 4 dimensions or so, we cannot visualize. So, for the purpose of visualization, this particular higher dimensional data are to be mapped to the lower dimension, say in 2 dimensions, say x and y . Now, if I want to map, these higher dimensional data to the lower dimension for the purpose of visualization, we cannot do actually 1 : 1 linear mapping. So, we will have to go for some sort of nonlinear mapping.

Now, these higher dimensional data, we are going to map to the lower dimension for the purpose of visualization; that means, we want to see the relative position of the different points, which are there in the higher dimension, so these points, we are going to see in the lower dimension. Now, this technique is known as actually the visualization technique and or the dimensionality reduction technique.

Now, if you see the literature there exist a large number of techniques for this dimensionality reduction. Now, these dimensionality reduction or visualization techniques can be classified into two groups. So, we have got the dimensionality reduction techniques, say DRTs and these techniques can be classified into two groups; one is called the distance preserving tools and we have got actually the topology preserving tool.

Now, let us try to find out the difference between the distance preserving tool and topology preserving tool. So, by distance preserving tool, actually what do you mean? So, in the higher dimension, if I consider two points, one point is here, another point is here, so we can find out the Euclidean distance between them. Supposing that this is point i and this is your point j and this particular distance is nothing, but say d_{ij} .

So, in distance preserving technique actually, we consider only the Euclidean distance between them, but we do not consider the relative position of the j th point with respect to the i th point. Whether the j -th point is towards the left or towards the right or towards the top or towards the bottom with respect to this i -th point that is not considered in the distance preserving tool.

But, in topology preserving, actually we are going to maintain the topology; that means, in the higher dimension, if the j -th point is towards the top with respect to the i -th, in lower dimension, I am just going to map this point at this particular point. So, the relative position of these particular points, the two points, those things will be maintained in the lower dimension and that means, in topology preserving, not only the distance, but we try to maintain the relative position of a particular point, with respect to another point.

Now, this Self-Organizing Map or SOM, that is actually a very efficient tool for the topology preserving. Now, actually, I should mention one thing that in human brain, we use this type of network very frequently and that is why, actually we can remember the topology. For example, sitting at a particular place, we can say, for example, starting from here, the road towards another city, so what is the direction, how does it go from one city to another city?

So, we can imagine that we have got the topology preserving tool that is nothing, but self-organizing map in our brain. Now, this particular self-organizing map can also be used as a very efficient clustering tool. Now, in this course, we have already discussed the working principle of a few clustering tools like Fuzzy C-means clustering, entropy-based clustering, in detail.

Now, the similar type of problem can also be given to this self-organizing map and this self-organizing map is actually a very efficient clustering tool. And, as I mentioned that here, in this particular network, we use the principle of unsupervised and competitive learning. So, here, we do not use the concept of supervised learning, which we have already discussed.

Now, let us see, how to use the concept of this unsupervised learning or the competitive learning in this particular self-organizing map. Now, if I see the main task of this particular network, that is as follows. Supposing that, I have got some higher dimensional data, a large number of data point, for example, say I have got say one, this is the i -th one and this is the n -th one.

(Refer Slide Time: 08:12)

Self-Organizing Map (SOM)/ Kohonen Network

- Proposed by T. Kohonen before 1995
- It can be used as *visualization technique* or *dimensionality reduction technique* (topology preserving tool)
- It can be used as a *clustering algorithm* also
- It works based on *unsupervised* and *competitive learning*

Self-Organizing Map

So, I have got say capital N number of data points, in the higher dimension, say these are in L dimensions L -D and these particular data will be passed through the input layer of this particular network and our aim is to map these higher dimensional data, that is the L dimensional data to the lower dimensional layer and that is nothing, but the competition layer for the purpose of visualization. So, what they do is, each of these particular N data points in the higher dimension, we are going to map it to the lower dimension for the purpose of visualization.

So, although this input layer is in L dimensions, this particular competition layer or this is nothing, but the output layer should be either in say 2 dimensions or it could be say in 3 dimensions, so, this could be your the 3 dimensions.

Because we can visualize only up to 2 dimensions or say only up to 3 dimensions. So, these particular data points are to be mapped to the lower dimension and that is nothing, but the output layer or the competition layer. Now, we are going to discuss, in details, how can you do? So, this type of mapping from the higher dimension to the lower dimension by maintaining the topological information of these particular the data points. So, these things actually, we are going to discuss in much more details.

(Refer Slide Time: 09:48)

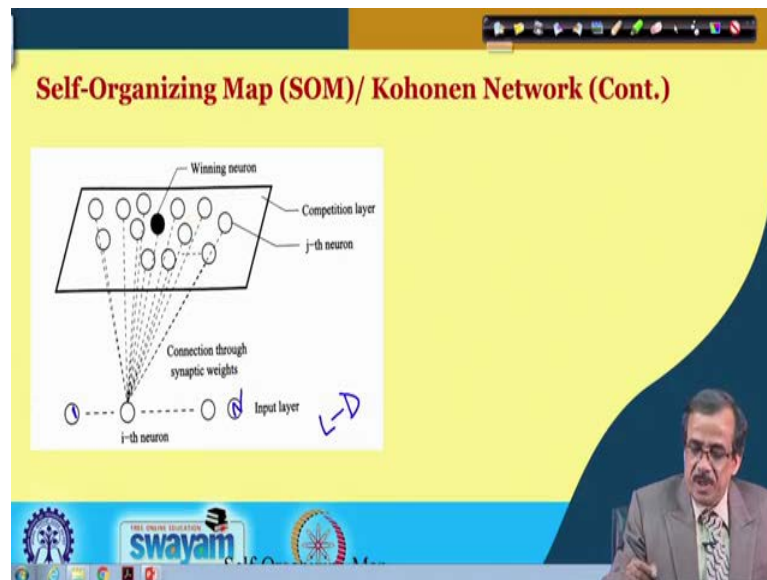
- Non-linear generalization of principal component analysis PCA
- Consists of two layers: Input layer and competition layer
- On competition layer, there are three basic operations, namely competition, cooperation and updating

Now, the self-organizing map can also be called actually nonlinear generalization of principal component analysis. You might have heard about the Principal Component Analysis, that is nothing, but PCA algorithm, now this PCA is actually a very efficient tool for linear mapping, 1 : 1 mapping, but as we are doing the mapping from higher dimension to lower dimension, we cannot think of the linear mapping, so we will have to go for actually the non-linear mapping.

And, in this non-linear mapping, we will not be getting this 1 : 1 mapping, there will be some in-accuracy and this in-accuracy, we will have to actually accommodate. Now, here if you see, the self-organizing map consists of two layers, for example, say we have got the input layer and the competition layer, this is nothing, but a two layer network. And, on this particular competition layer actually, there will be three basic operations and these operations, we are going to discuss it details.

These are nothing, but the competition, then there will be cooperation and after that, there will be updating and through this competition cooperation and updating, actually this particular network is going to do the mapping from the higher dimension to the lower dimension. Now, let us see how does it work? So, let us first concentrate on the competition. Now, the purpose of this particular competition is to declare a winner. Now, let us see, how to declare and how to determine that particular the winner?

(Refer Slide Time: 11:44)



Now, this is the network; this is the network although I have shown it here. So, this is the input layer and the competition layer and I have already drawn this particular competition layer, but for the time being, let me assume that this particular competition layer is actually absent. So, this competition layer, actually these type of neurons are (the way of shown it here) are absent for the time being, say.

Now, what is our aim? Let me repeat we have got capital number of data. So, starting from 1 up to say capital N data points, we have on the input layer and these are in L dimensions, that is your L-D or say the higher dimension or any other dimension say. Now, these particular data points are to be mapped to the lower dimension. Now, let us see, how can we use the principle of this particular competition, now let us see the principle of this competition first.

(Refer Slide Time: 12:53)

Competition

Let us assume that there are N points (neurons) in the input layer and each point has m dimension

$$X_i = [x_{i1}, x_{i2}, \dots, x_{im}]^T,$$

where $i = 1, 2, \dots, N$

Let the synaptic weight vector between input neuron i and neuron j lying in the competition layer is denoted by

$$W_j^i = [w_{j1}^i, w_{j2}^i, \dots, w_{jm}^i]^T,$$

where $j = 1, 2, \dots, N$

The slide also features a 'swayam' logo and a small image of a person in the bottom right corner.

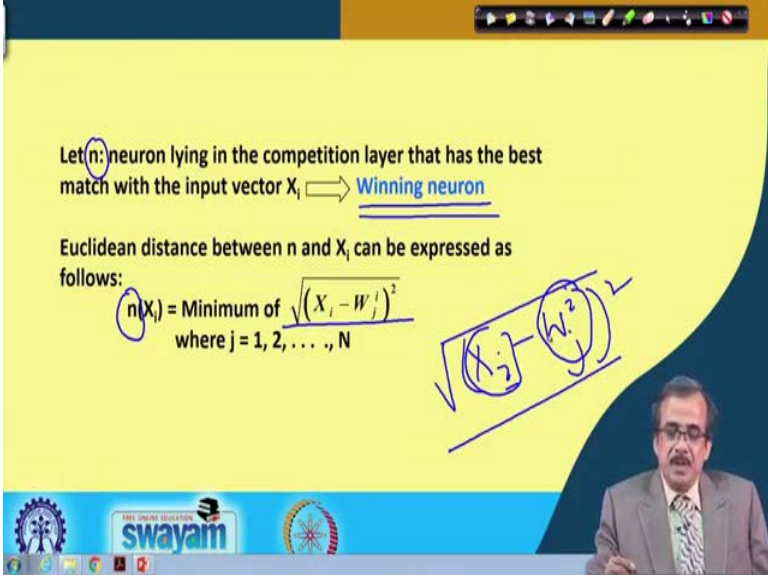
Now, here, as I told that we are going to map a particular data points, say i -th data point lying on the input layer. Now, this particular i -th data point, supposing that I am considering here m dimensional point. So, if it is having m dimensions, so to represent this particular X_i , so, I will have to use x_{i1} , x_{i2} up to x_{im} . So, I have got m number of numerical values, where i varies from 1, 2 up to capital N . So, this N is actually the total number of data points and each point is having say small m dimensions, now, how to carry out this particular competition.

Now, to carry out the competition actually, what we do is, we generate, at random, some correcting weights between this input neuron i and the neuron j lying on the competition layer or this output layer. Now, what do you do is, we generate some connecting weight or synaptic weight that is denoted by W_{ji} . Now, if you want to generate so this particular W_{ji} , the first thing you will have to do is, you will have to decide its dimension. Now, let me assume that once again, this is having m dimensions, that means, the same dimensions of your this input data.

Now, what you do is, so this W_{ji} to represent, you will have to use small m number of numerical values, that is nothing, but w_{j1}^i , w_{j2}^i and the last is your w_{jm}^i and j varies from 1, 2 up to N , that is nothing, but the total number of data points lying on the input layer.

Now, once you have got these particular the W values, which are generated, now very easily I can find out what should be the Euclidean distance between your X_i and your this particular W_i and through this competition actually, what you are going to do is, we are trying to find out that the value of W , which is the closest to this particular X_i .

(Refer Slide Time: 15:33)



Let n : neuron lying in the competition layer that has the best match with the input vector $X_i \Rightarrow$ Winning neuron

Euclidean distance between n and X_i can be expressed as follows:

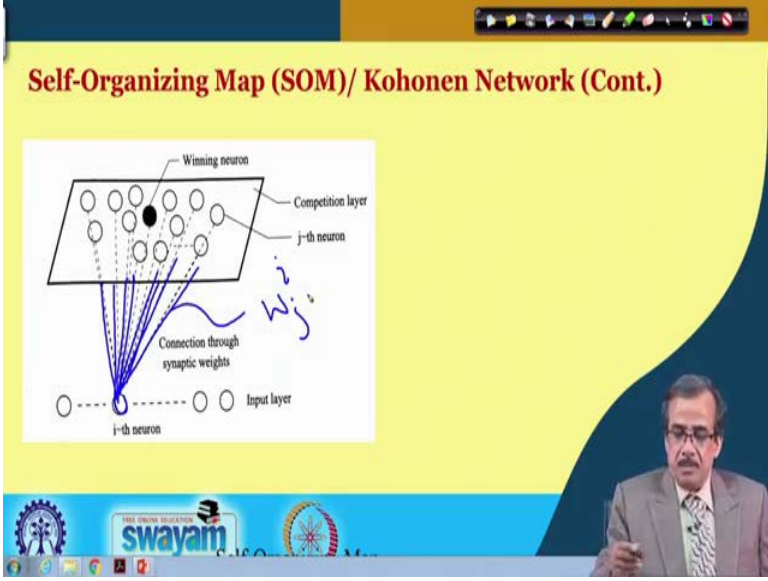
$$n(X_i) = \text{Minimum of } \sqrt{(X_i - W_j)^2}$$

where $j = 1, 2, \dots, N$

Handwritten notes on the slide include a circled 'n' and a diagram showing a vector X_i and a weight vector W_j with a distance line between them.

Now, before I go for that particular calculation, I am just going to show you on the schematic view. Let us suppose, if this is the i -th data point, which I am going to map to the lower dimension,

(Refer Slide Time: 15:46)



Self-Organizing Map (SOM)/ Kohonen Network (Cont.)

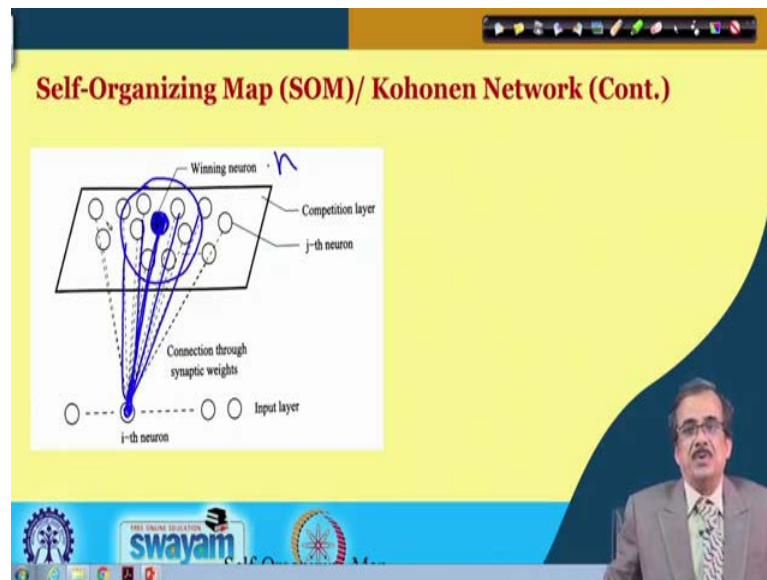
The diagram illustrates the architecture of a Self-Organizing Map (SOM) or Kohonen Network. It shows an **Input layer** at the bottom with a node labeled i -th neuron. Above it is the **Competition layer** with multiple nodes, one of which is highlighted as the **Winning neuron**. Dashed lines represent **Connections through synaptic weights** between the input neuron and the neurons in the competition layer. A handwritten note W_{ij} is next to one of these connections.

what I do? We actually generate all such W values. So, these are all connecting weights or the W values and these are actually your say W_j^i and these actually the connecting weights are generated at random using the random number generator, supposing that in the normalized scale, between say 0 to 1. Now, if this is the situation, then very easily, I can find out the Euclidean distance between the i -th data point lying on the input layer and capital N number of W values, each W is having small m dimension.

Now, how to do this? So, let me explain that now as I mentioned that we are going to determine the Euclidean distance between your X_i and W_j^i and these particular Euclidean distance is nothing, but $\sqrt{(X_i - W_j^i)^2}$. So, this is nothing, but the Euclidean distance between X_i and W_j^i . Now, corresponding to this particular i -th data point, we have generated how many W values? Capital N number of W values. So, how many such Euclidean distances are possible we have got capital N number of Euclidean distance values and out of these capital N number of Euclidean distance values, what I do is, we try to find out which one is the closest, which one is having the minimum Euclidean distance values and that particular W which is having the minimum Euclidean distance is actually declared as the winning neuron or the winning connecting weight. So, mathematically, the winning neuron or the winning connecting weight is denoted by small n , and your the Euclidean distance between n and X_i is expressed like this, and so this n is consider as the winner; that means, it is having the minimum of these Euclidean distance values.

Now, let me repeat, out of these Euclidean distance values, we try to locate that particular W_j^i , which is the closest to your X_i and which will give rise to the minimum Euclidean distance value and that particular W_j^i will be declared as the winner. Now, the purpose of this particular competition is actually to declare the winner of this competition, now if you just see on your plot.

(Refer Slide Time: 18:53)



So, here, once again corresponding to the i -th one, I am trying to find out the winner, supposing that, this particular connecting weight is nothing, but the winner. So, this indicates actually the winner neuron and that is denoted by your small n . So, small n is nothing, but the winning neuron, ok. Now, this is the winner, now surrounding this particular winner, there will be some other connecting weights.

Now, for the time being, these particular neurons are not drawn, it is simply the connecting weights. So, these are simply the connecting weights, the neurons are not drawn now. So, surrounding this particular connecting weight, we have got a number of other connecting weights in this neighborhood, ok. So, let me repeat, the purpose of this particular competition is to declare that winner and once that particular winner we have got, now we can enter the cooperation stage.

(Refer Slide Time: 20:00)

Cooperation

Surrounding a winning neuron, a neighborhood of excited neurons is defined for cooperation in order to update their synaptic weights

$$h_{j,n(x_i)}(t) = \exp\left(-\frac{d_{j,n(x_i)}^2}{2\sigma_t^2}\right),$$

where $t = 0, 1, 2, \dots$

$d_{j,n(x_i)}$: Lateral distance between the winning neuron n and excited neuron j

σ_t : Standard deviation at t -th iteration

$\sigma_t = \sigma_0 \exp(-t/\tau)$

σ_0 : Initial value of standard deviation

τ : Predefined number of maximum iterations

Handwritten notes on the slide include: σ_t (circled), $t = 1000$ (circled), and σ_t (written next to the Gaussian curve).

The next stage is actually the cooperation and here, in this particular stage, we have already decided the winner through competition and surrounding that particular winner, there will be some excited connecting weights or excited neurons. So, let me just draw it here; now here, if you see this particular surrounding that is expressed with the help of mathematical expression, that is nothing, but the Gaussian distribution.

Now, if I just draw this particular equation of one Gaussian distribution, it will look like this, so this is actually the Gaussian distribution and it is having your the standard deviation that is nothing but is your σ_t and it is having the mean. So, mean is nothing, but the property of these particular the winners.

So, the mean properties are decided by the winners property, which have been declared the winner in that particular competition and there will be a Gaussian distribution, and the nature of the Gaussian distribution will be decided by the standard deviation, that is, σ_t ; t means that t -th iteration. Now, if you see that $h_{j,n(x_i)}$, so j is actually your excited neuron or the excited connecting weight and small n is nothing, but the winner.

So, the neighborhood function between the j -th excited connecting weight and the winner

small n at t -th iteration is nothing, but $h_{j,n(x_i)}(t) = \exp\left(-\frac{d_{j,n(x_i)}^2}{2\sigma_t^2}\right)$. Now, this particular $d_{j,n(x_i)}$,

$n(x_i)$ is nothing, but the lateral distance between the winning neuron n and the excited neuron j . So, very easily, you can find out this particular Euclidean distance.

And, once you have got these, now let us concentrate on these σ_t . So, this σ or the standard deviation, it is not kept fixed and it is actually a variable and it will vary from your iteration to iteration. Now, here we have written this particular $\sigma_t = \sigma_0 \exp(-\frac{t}{\tau})$.

Now, this particular τ is nothing, but pre defined number of maximum iterations, supposing that the τ is kept equal to say 1000 or something like this, so τ you say equal to 1000.

Now, so this is actually the fixed number, now as iteration proceeds, what will happen to this σ_t ? So, as iteration proceeds, small t is going to increase now as small t increases, what will happen to this particular σ_t . So, this particular σ_t is going to be reduced. And, if σ_t reduces, what will happen to the nature? Now, if you see this particular nature, if σ_t reduces, I will be getting some sort of steeper distribution of these particular Gaussian.

So, as iteration proceeds, there is a possibility that I will be getting say this type of steeper distribution and as iteration proceeds, I may get even steeper distribution corresponding to these, so this type of steeper distributions I will be getting. Now, if I take the plan view corresponding to the first the Gaussian. So, there is a possibility that this indicates the plan one, plan view for the original, as the iteration proceeds, so it is going to be reduced something like this and there is a possibility that this particular neighborhood is going to be reduced and will be getting actually a lot of interactions and through this particular interaction between the mean properties, that is a winner and you are the excited connecting weights or the excited neurons, there will be a chance of updating of both the things.

Now, this is the almost similar to the situation like supposing that, say in one institute there are a large number of professors and around under each professor, a large number of students are working. Now, these students are all excited neurons or excited connecting weights and as if the professor is having the mean properties and professor is the winner. Now, there will be lot of interactions between the professor and the students

and through this particular interaction, there is a possibility that the students are going to update their knowledge level and at the same time, the professor is also going to learn a few new things.

(Refer Slide Time: 25:58)

Updating

Synaptic weights of the winning neuron and excited neurons are updated as follows:

$$W_j^i(t+1) = W_j^i(t) + \eta(t)h_{j,n(x_i)}(t)[X_i - W_j^i(t)],$$

$\eta(t)$: learning rate (0.0, 1.0)

Input

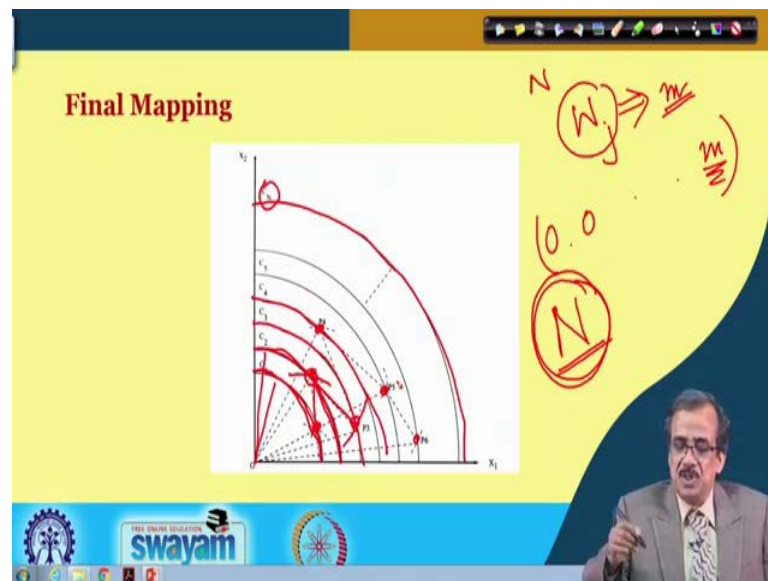
So, through this particular interaction, both the professor as well as the students are going to learn and there will be some sort of updating. Now, the principle of updating is very simple. So, what we do is we try to update say $W_j^i(t+1) = W_j^i(t) + \eta(t)h_{j,n(x_i)}(t)[X_i - W_j^i(t)]$.

And, this particular learning rate is going to vary in the range of say 0 to 1 and through this particular interaction, both the excited neuron as well as the winner are going to be updated, their connecting weights are going to be updated through a large number of iterations.

So, corresponding to that particular i-th data point, if you remember, like on the input side we have got a number of neurons, so this is the first one, this is actually the i-th one and this is your N-th one. So, corresponding to this particular ith one, so actually, we are going to get the updated one and ultimately, corresponding to this, I will be getting finally, one connecting weight something like this, that is the modified winner. Now, you repeat the same process for the remaining N minus 1 data points lying on the input layer, so this is the input layer. So, lying on the input layer, you repeat the process inside a for loop of the computer program.

So, corresponding to each of these particular data points, in the higher dimension, I will be getting a actually one modified winner W . So, for each of the data points actually, I will be getting these particular modified connecting weights. Now, once you have got this type of modified connecting weights, now we will have to do the mapping. Now, let us see, how to do this particular mapping.

(Refer Slide Time: 28:38)



Now, let us try to actually understand the situation. So, on the input side or on the input layer, we have got capital N number of data points and for each of the data points, in the higher dimension, I have got the modified, winner connecting weights. So, I have got, how many such W values, once again I have got capital N number of W values.

So, we have got capital N number of W values, so what I can do is, now this W is having the m dimensions. So, what you can do is, if I consider that m dimensional space its origin is, what I have got actually small m number of 0s. So, we have got small m number of 0s on the origin and what we can do is, we have got capital N number of W values corresponding to capital N number of neurons lying on the input layer.

Now, what you can do is, starting from the origin, I can find out the Euclidean distance of all the W values, all capital N number of W values. Then, how many Euclidean distance values, we are going to get? So, we are going to get actually capital N number of Euclidean distance values. Let me repeat, starting from the origin in m dimensions like I have got 0 0 0 small m number of 0 at the origin.

So, I can find out, I can calculate the equilibrium distance of capital N number of W's and that means, I can find out capital N number of Euclidean distance values and once I have got capital N number of the Euclidean distance values, we do the sorting in the ascending order.

So, the W, which is the closest to the origin will be considered first and we are going to do the sorting in the ascending order and once you have got that particular information of Euclidean distance in the ascending order, what you can do is, we can consider that particular w, which is the closest to the origin and its Euclidean distance we can consider as radius and we can draw one circular arc something like this, that is denoted by c_1 .

And, the next Euclidean distance value, I can use as your the radius, I can draw another circular arc. Similarly, I am drawing another circular arc and how many such circular arcs, I will have to draw? I will have to draw capital N number of circular arcs. So, I am just going to draw capital N number of circular arcs. Now, let me concentrate on the first circular arc, you take a point at random lying on the first circle so let us consider this particular point.

And, now, I know the Euclidean distance between these particular Ws, so this corresponds to one W, this corresponds to another W, the connecting weight. So, I know the Euclidean distance between this W and that particular W and its numerical value I can calculate and considering that as the radius, I can draw another circular arc and supposing that I am drawing this particular arc.

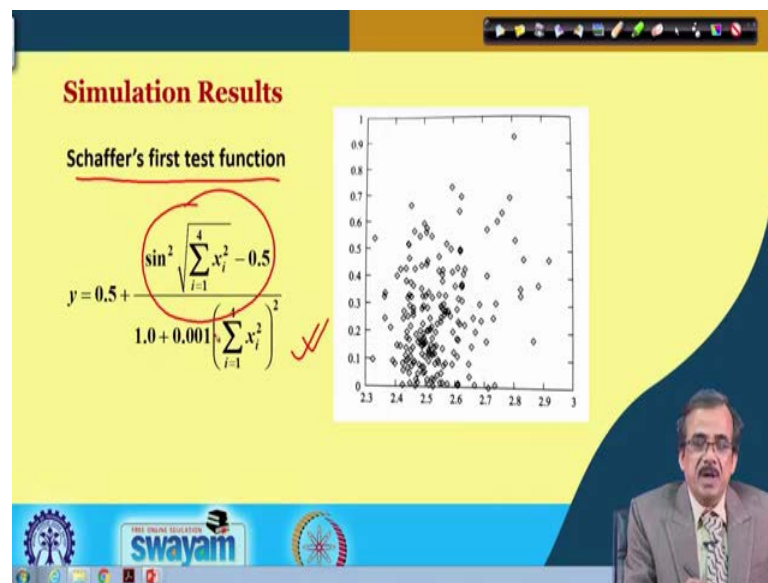
Now, once you have got, so this is another point, now considering this at the center and considering your the next W, I can find out the Euclidean distance and these particular Euclidean distance we can consider as the radius and I can draw one circular arc here. So, I will be getting another intersection point, following the same procedure, I will be getting another intersection; another intersection; another intersection, and so on, and once you have got all such intersection points, those are nothing, but the points in two dimensions or the lower dimension. So, these higher dimensional data, now, we are in a position to map to the lower dimension like say two dimensions for the purpose of visualization.

Now, if you just go back to your diagram, which we have considered for the self-organizing map, now we will be able to explain these , like corresponding to these

supposing that I have got only one winner. Similarly, corresponding to another data point might be another winner and modified winner, say corresponding to this might be another, corresponding to this might be another. So, capital N number of data points, lying on the input layer will be mapped to the lower dimension on the competition layer for the purpose of visualization.

So, this is the way actually, from the higher dimension, we can do the mapping to the lower dimension for the purpose of visualization. So, now, we will be in a position to draw all such neurons here, and each neuron indicates a particular point lying on this input layer. So, I will be getting here capital N number of neurons and each neuron is going to represent a particular neuron in the higher dimensional input layer. Now, this is the way, actually it works.

(Refer Slide Time: 34:48)



Now, I can show one example like very simple example like this is actually one test function, that is called the Schaffer's test function. So, this Schaffer's test function, if you see the mathematical formulation, so this is the mathematical expression for Schaffer's test function and here, you see

$$y = 0.5 + \frac{\sin^2 \sqrt{\sum_{i=1}^4 x_i^2}}{1.0 + 0.001 \left(\sum_{i=1}^4 x_i^2 \right)^2} - \frac{0.5}{1.0 + 0.001 \left(\sum_{i=1}^4 x_i^2 \right)^2}$$

expression and here, you see i varies from 1 to 4; that means, this is in 5 dimensions.

Now, these 5 dimensional data, we cannot visualize because we can visualize only up to 3 dimensions. Now, what we do? We generate 1000 data points at random, lying on the surface of this particular test function and these 1000 data points are mapped to the 2 dimensions for the purpose of visualization using the self-organizing map.

So, in the higher dimensions, if there are 1000 data points in the lower dimension also, it will be getting 1000 data points and here, the topological information will be kept unaltered or intact. And, here, these particular data points are well-distributed, so there is an ease for visualization. So, very easily, we can visualize these data points and their relative positions or their topology, we can visualize very easily.

Thank you.