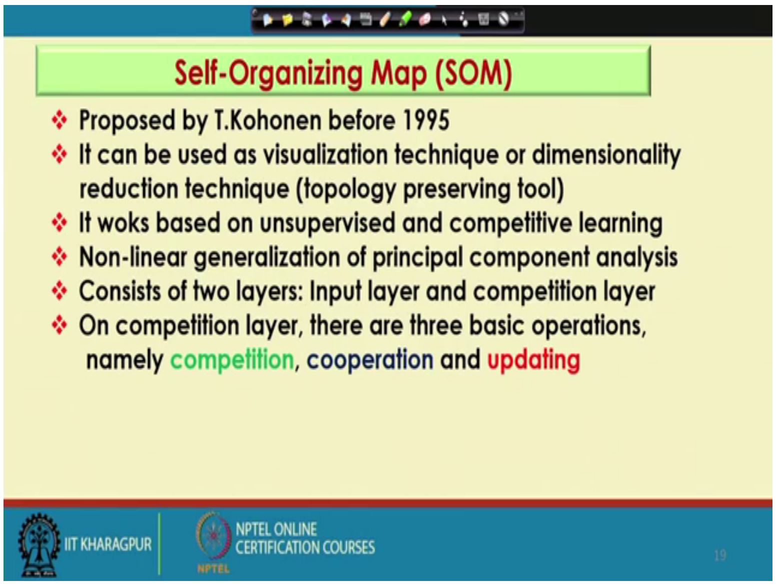
Traditional and Non-Traditional Optimization Tools

Prof. D. K. Pratihar
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture - 15
Faster Genetic Algorithms (Contd.)

Now, I am just going to start with the working principle of another very efficient or non-linear mapping tool which is popularly known as the self organizing map in short SOM.

(Refer Slide Time: 00:31)



Self-Organizing Map (SOM)

- Proposed by T.Kohonen before 1995
- It can be used as visualization technique or dimensionality reduction technique (topology preserving tool)
- It woks based on unsupervised and competitive learning
- Non-linear generalization of principal component analysis
- Consists of two layers: Input layer and competition layer
- On competition layer, there are three basic operations, namely competition, cooperation and updating

Now, this is actually a special type of neural network; now using this particular neural network the self organizing map, I can also map the higher dimensional data to the lower dimension approximately and this is one non-linear mapping tool. Now this mapping tools are actually also known as the dimensionality reduction technique.

Now, the self organizing map was proposed by Kohonen before 1995 around 94; actually the idea came first. Now it can be used as a visualization technique or the dimensionality reduction technique. And here let me mention that this is a topology preserving tool; now unlike the other two methods which have already discussed. So, this method is the topology preserving tool; that means, here not only the Euclidean distance we try to keep intact, but also the relative position of the second point with respect to first, we also try to keep intact.

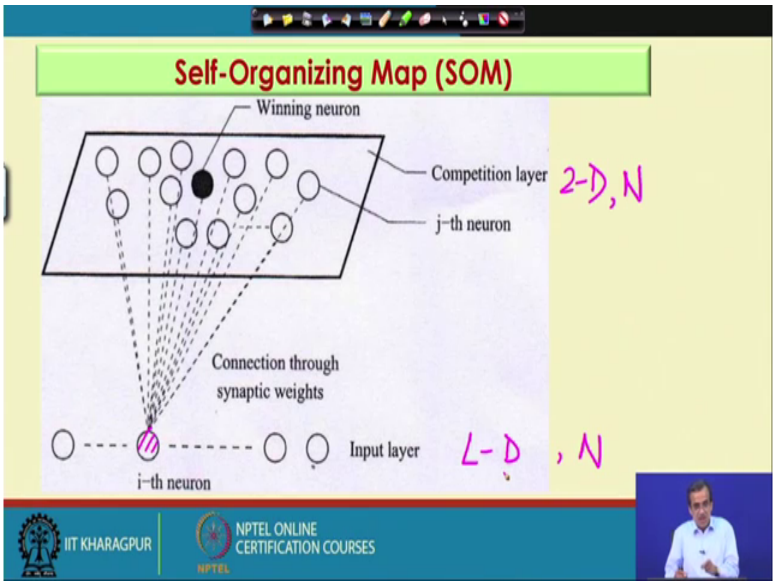
Let me take a very simple example; now suppose that we are going to consider the distance between Calcutta and Delhi. So, the (Refer Time: 02:06) the Euclidean distance will have some numerical value. And in distance preserving technique, we consider only the numerical value.

But here in topology preserving tool not only this numerical value of the distance, but another information we consider that is nothing, but the relative position of Delhi with respect to Calcutta. So, Delhi is towards not with respect to Calcutta; so that particular information is also kept intact in this type of mapping and that is why this is known as a topology preserving tool.

Now, it works based on unsupervised and competitive learning; now this is a unsupervised learning. Because here there is no known input output relationship and if there is no such input output relationship. So, we will have to take the help of competitive learning. Now how to implement the competitive learning? That I am going to discuss in details. Now this is nothing, but non-linear generalization of principal component analysis PCA.

Now, principal component analysis is a very efficient tool for linear mapping, but here we are going for non-linear mapping. And for this non-linear mapping, we are going to use self organizing map and which is nothing, but a non-linear generalization of this linear mapping tool; that is the principal component analysis. It consists of two layers input layer and competition layer; now on the competition layer there will be three basic operations like competition, cooperation and updating. So, how to implement this competition, cooperation and updating? So, that I am going to discuss in details.

(Refer Slide Time: 04:23)


Self-Organizing Map (SOM)

Now, this shows actually the schematic view of this particular the self organizing map. Now let us try to understand what is there? Now suppose that on the input layer; that is in higher dimension say L dimension. So, this input layer is an higher dimension say L D and we have got the competition layer say that is in say lower dimension say 2 D.

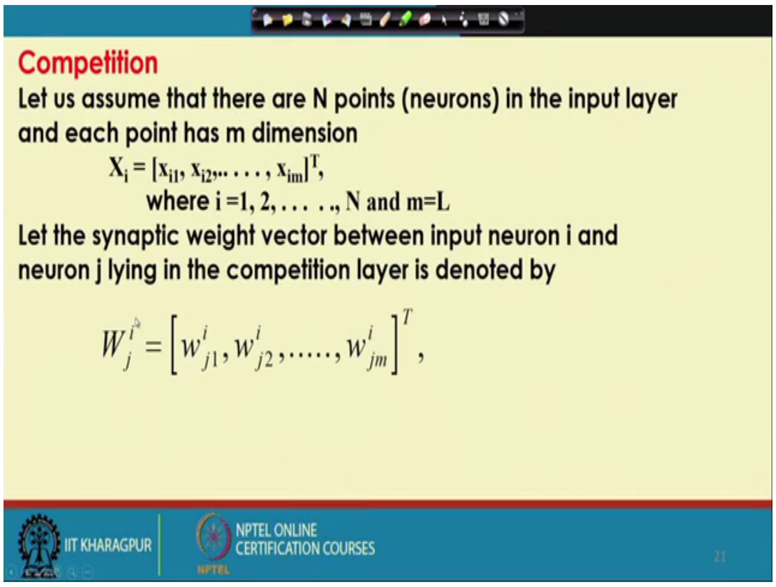
So, on L D the higher dimension we have got a large number of data points say capital N number of data points. So, our aim is to do the mapping from this higher dimension to the lower dimension; that means, on the lower dimension I will have to find out capital N number of data points.

Now, how to do it that; I am going to discuss, now suppose in that out of this capital N data points; lying on the input layer, I am just concentrating on a particular data point that is the i-th data point. Now i-th data point is represented with the help of i-th neuron and neuron we know; in biological nervous system neuron is nothing, but the unit; unit of the biological nervous system. So, here artificially we copy that particular the biological neuron; now let me assume that so this particular circle is going to represent a particular neuron and which is also going to represent the i-th data point.

Now, similarly we have got capital N number of data points; that means, capital N number of neurons here. Now for the time being, let me concentrate on this particular the i-th neuron or the i-th data point; now how to represent this i-th data point? Once again I will have to consider the there dimension if it is L D. So, there will be capital L number of numerical values to represent this particular the i-th input data point. And now I will have to find out the corresponding data point in the lower dimension that is in the competition layer.

Now, how to get it? To get it actually what we will have to do is; first we will have to represent; so, this particular data point in higher dimension.

(Refer Slide Time: 06:58)



Competition

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

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

where $i = 1, 2, \ldots, N$ and $m = L$

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

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

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now let me start with the competition; so, here as I told that a particular data point that is X i is represented by capital L or small m; here I have kept small m is equals to capital L. So, this small m is nothing, but capital L; so, e particular data point; the i-th data point is represented by small m number of numerical values or capital L number of numerical values. So, X i is nothing, but X i 1 comma X i 2; the last term is X i m transpose or I varies from 1, 2 up to N I am here as I told I have assumed small m is equals to capital L.

This is how to represent the data point on the input layer that is in a higher dimension. Now what you do is I am just going to find out what should be the corresponding data point in the lower dimension that is on the competition layer? Now to do this actually what you do is; corresponding to this particular the data point; what we do? We generate some synaptic weights denoted by W j i. Now W j i is nothing, but the synaptic weight or the connecting weight between the input i and the neuron lying on the competition layer and that particular neuron or the data point is nothing, but j.

So, initially we do not know where the j is. So, what you do is we generate W j i that is a connecting weight between the input neuron i and the data point j which is lying on the competition layer and what you do is we generate a large number of connecting weights. So, what is the number?

Now generally what you do is if there are 1000 data points to be mapped corresponding to a particular data point. So, we generally generate 1000 this type of connecting weight that is W j i. Now to represent a particular W j i once again; I will have to take the help of small m numerical values like w j y i 1 comma W j i 2 comma the last term is W j i m. So, all such connecting weight values will be generated at random using the random number generator.

Now, let me let me go back to the earlier slide where we showed this particular the schematic view. Now let me once again concentrate here; so what you do is corresponding to this particular input. So, we generate a large number of connecting weights denoted by W j i and as I told we generally consider 1000; this W j i values and to represent a particular W j i, we use small m number of numerical values lying between 0 and 1; supposing that we have generated 1000 W j i values.

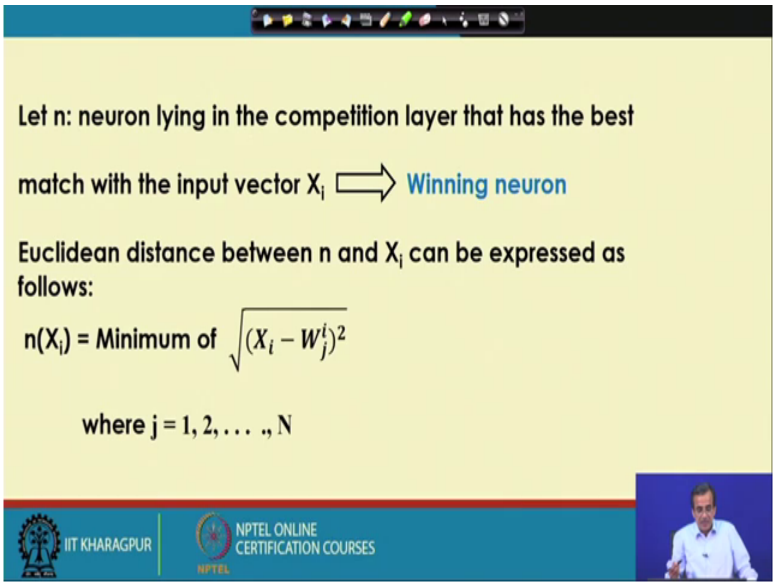
Now, once I have generated those W j i values at random; now I can do is, I can determine the Euclidean distance between the i-th data point and all such W j i values. So, how many Euclidean distance values will be getting? I will be getting 1000 Euclidean distance values. And that is nothing, but the Euclidean distance between the i-th input point and the W j i that connecting weights.

So, I will be getting like 1000 Euclidean distance values and we compare their numerical values and try to find out one point or one connecting weight, which is closest to this particular the i-th data point. And the data point or the connecting weight which is found to be closest to the i-th data point is declared as the winning point or the winning neuron.

Suppose this is the winning neuron or the winning or data point in lower dimension; now it is not in lower dimension till now it is in higher dimension. So, corresponding to this i-th data point; I can find out this particular winner; that means, the winner W j i values. Now once you have got this particular winner; so, what you do is the competition that particular operation of competition is actually over. And when this competition is over;

so we are able to declare the winner. So, once you have got this particular winner; now there will be some sort of updating.

(Refer Slide Time: 12:37)



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

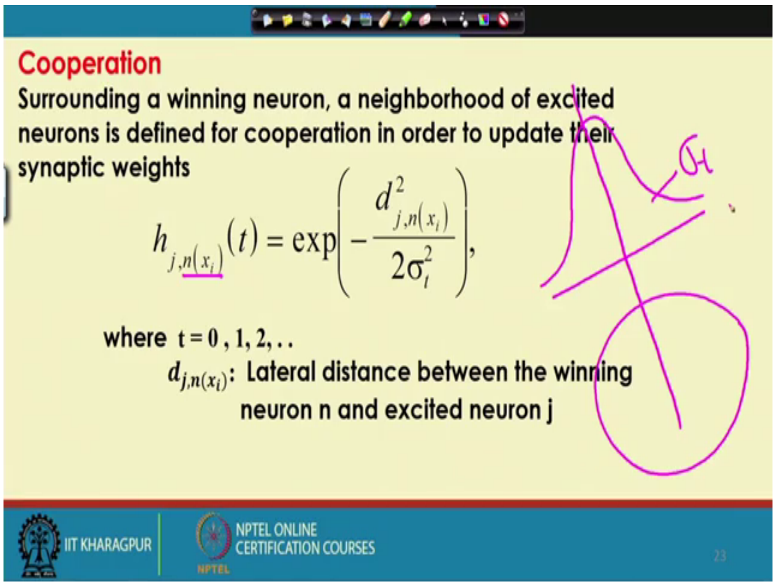
Euclidean distance between n and X_i can be expressed as follows:

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

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

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

(Refer Slide Time: 12:45)



Cooperation

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

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

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

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

σ_t
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
23

Now, there will be a corporation; now what I do is corresponding to the i-th data point on the input layer. So, I have got one winner and that is nothing, but winner connecting weight and suppose that we have got the winner connecting weight here and surrounding that winner. So, we are going to design one neighborhood and generally we take the help of some sort of Gaussian distribution to represent the winner and its neighbourhood.

So, surrounding the winning neuron actually we try to find out a neighborhood of excited neuron. And there will be some interaction between the winner and this excited neighbourhood and there will be updating of the synaptic weights or the connecting weights.

Now, how to implement that? To implement that actually what we do is; we take the help of one Gaussian distribution as I told. Now here; so for this Gaussian distribution actually we try to represent in this particular form that is h j n x i. So, this n x i is the winner; j indicates a never of that particular winner and this h j n x i is used to represent that Gaussian distribution of this particular the neighbourhood.

Now this is nothing, but exponential minus d j n x i square divided by 2 sigma t square. Now what is this d j n x i? So, this d j n x i is a lateral distance between the winner and the excited neuron j.

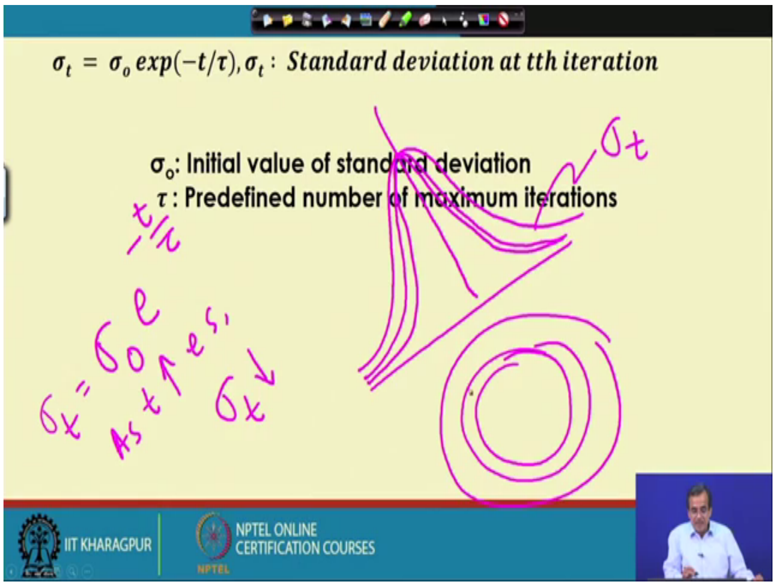
Now, let me just draw it here; now as I told that we are going to consider some Gaussian distribution to represent this neighborhood. Now suppose in that I have got a Gaussian distribution something like this. So, this is the Gaussian distribution and it has got the mean and standard deviation; say this mean actually the mean properties are properties of that particular winner.

And this Gaussian distribution will have some sigma t; that is standard deviation. So, once I know the mean properties and standard deviation, I can draw this particular the Gaussian distribution which is nothing, but the neighbourhood.

Now, as I told that the mean properties will be decided by the properties of the winner or the properties of the leader. And if I take the plan view; so might be I will be getting this type of plan view for this the Gaussian distribution. Now, this is almost similar to the situation that in one department; there are a few professors working in different areas, as if each of the professors are widen and leader.

And surrounding the professor there will be a few students working for there say PHD's or M Tech. Now; so, surrounding this professor, so there will be a few followers or the students and this will generate one Gaussian distribution sort of thing. It is almost similar to this type of situation; now what happens is this particular sigma t, we do not consider to be a constant; instead we vary this particular the sigma t. So, how to vary this particular sigma t? Now to vary this particular sigma t, actually what we do?

(Refer Slide Time: 16:54)



$\sigma_t = \sigma_o exp(-t/\tau), \sigma_t$: *Standard deviation at tth iteration*

σ_o: Initial value of standard deviation

τ : Predefined number of maximum iterations

$\sigma_t = \sigma_0 e^{-t/\tau}$

As $t \uparrow$ es, $\sigma_t \downarrow$

So, we take the help of one mathematical expression; that is sigma t is nothing, but is sigma naught exponential minus t divided by tau. So, this sigma t is nothing, but the standard deviation at t th iteration and this sigma naught is actually the initial value of the standard deviation; that is the fixed quantity. Now here if we write down that sigma t is nothing, but sigma naught exponential minus t divided by tau.

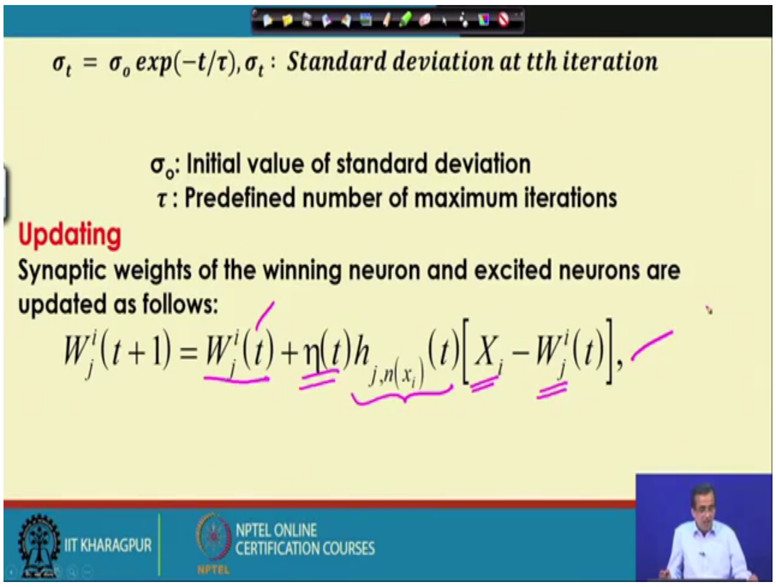
So, t is the iteration number; tau is the maximum number of iteration which is having the fixed value. So, as t increases; that means, as iteration proceeds what will happen to this particular sigma t? The sigma t is going to be reduced; so, initially there will be higher sigma t and with the number of iteration, the sigma t is going to be reduced. Now what does that mean?

Now if I draw that particular Gaussian distribution once again. So, I will be getting; so, this type of Gaussian distribution for the neighborhood and if I take the plan view might be it something like this. And this it has got a mean property that is the property of the professor; the leader and here we have got this particular sigma t.

Now, as I told with the number of iteration the sigma t is going to be reduced. So, might be in the next iteration; the Gaussian distribution could be something like this. And here accordingly the plan view will be something like this; now once again next iteration there is a possibility I will be getting; so, this type of sigma t. So, the plan view will be something like this; now what does it mean? It means that with the number of iteration the neighborhood is going to shrink and within the neighbourhood the excited members the excited students and the professor there will be lot of interaction. And through this particular interaction both the professor as well as the students are going to learn a lot.

So, both the students as well as the professors will be benefited through this particular the cooperation. So, this is actually the principle of cooperation; now this principle has been copied in the artificial way just to implement the cooperation in self organizing map. Now once is this particular cooperation is over. Now how to update the connecting weights or how to update there, the level of knowledge, so that I am going to discuss.

(Refer Slide Time: 19:55)



$\sigma_t = \sigma_0 exp(-t/\tau), \sigma_t$: *Standard deviation at tth iteration*

σ_0: Initial value of standard deviation

τ : Predefined number of maximum iterations

Updating

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

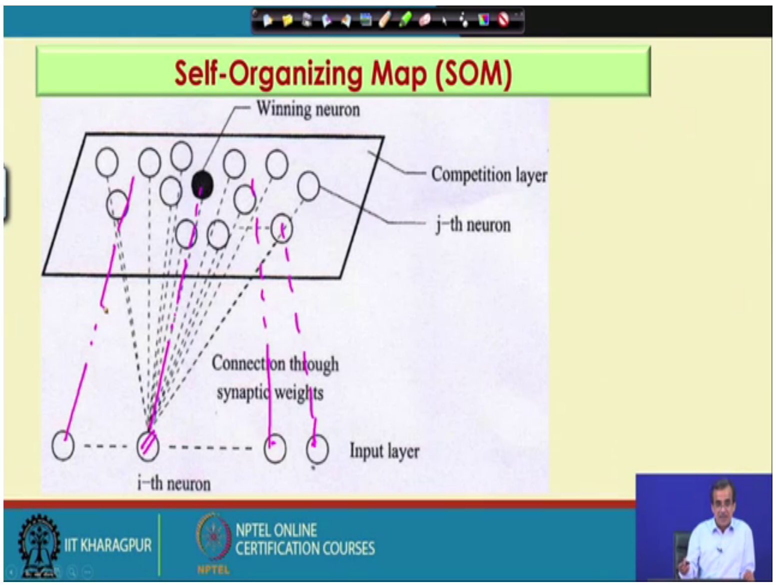
$$W_j^i(t+1) = \underline{W_j^i(t)} + \underline{\underline{\eta(t)}}\underbrace{h_{j,n(x_i)}(t)}\left[\underline{\underline{X_i}} - \underline{\underline{W_j^i(t)}}\right],$$

Now we go for the updating; now this is the rule for updating. Now if you see; so this W j i t plus 1; that means, what will happen to the connecting weight between the input X i and the point lying on the competition layer; that is the jth one.

So, this is the connecting weight; so, what will happen to the value at t plus 1 th iteration that depends on the value of the connecting weight at t th iteration plus eta t is the learning rate. Then comes h j n x i is the neighborhood function into X i minus W j i t; X i is nothing, but the input vector in higher dimension W j i is the connecting weight in higher dimension. So, this is actually the rule for updated and using this principle of updating both the professors as well as the students are going to update their knowledge level and both will be benefited.

So, this particular principle has been copied here; that means, if you follow these for a number of iterations, ultimately corresponding to a particular input data point X i; I will be getting only one updated W j i connecting weight. Now once again let us go back or to the schematic view. So, which we consider for the self organizing map; now this is the schematic view, so corresponding to this particular the i-th data point.

(Refer Slide Time: 21:56)



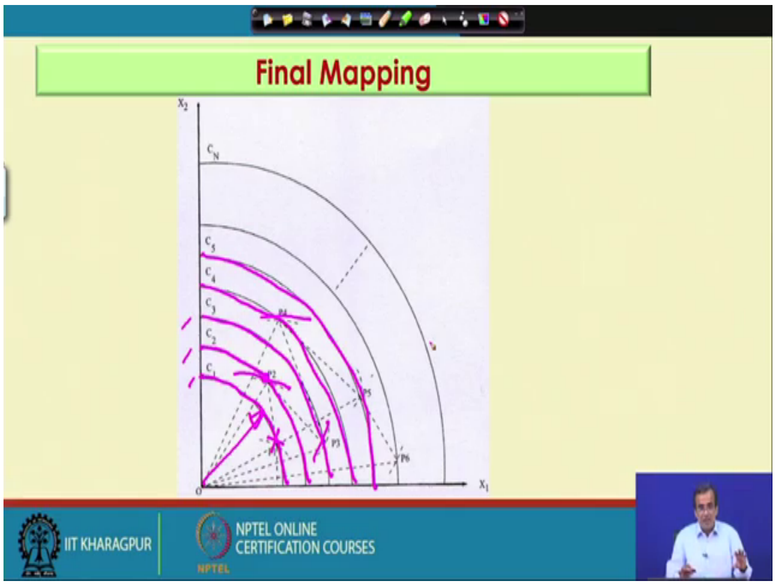
Self-Organizing Map (SOM)

So, I have got say one connecting weight say this is the connecting weight updated one. Now same principle you follow for all the remaining N minus 1 input data. So, corresponding to this you try to find out another connecting weight and its updating.

Corresponding to this another W j i value I will be getting; corresponding to this another connecting weight I will be getting. So, corresponding to capital N data points here, so I will be getting capital N connecting weights and its updated values; now what you do is. so, all such connecting weights are in higher dimension, in m dimension or capital L dimension.

Now I will have to go for mapping; so, how to do this particular the mapping? Now to implement the mapping; actually what you do is we follow a principle which I am going to discuss in details.

(Refer Slide Time: 23:08)


Final Mapping

Now, so till now corresponding to each of the data points. So, I have got one connecting weights the updated value for the connecting weights. Now we have got capital N number of data points; so, we have got capital N number of the connecting weights or the synaptic weights.

Now what you do is these synaptic weights are in m dimension or capital L dimension; that is the higher dimension. So, starting from the origin of this higher dimension; I can find out the Euclidean distance of all the connecting weights.

Now, we have got capital N number of connecting weights. So, I will be getting capital N number of Euclidean distance values and once you have got all such Euclidean distance values; now I can sort them in the ascending order. That means, the connecting weight which is having the minimum Euclidean distance value from its origin, will be considered first and so on.

So, what you do is we sorting all such connecting weights that is the W values in terms of their Euclidean distance values; counted from its origin in higher dimension. Now supposing that I am getting that particular order in ascending.

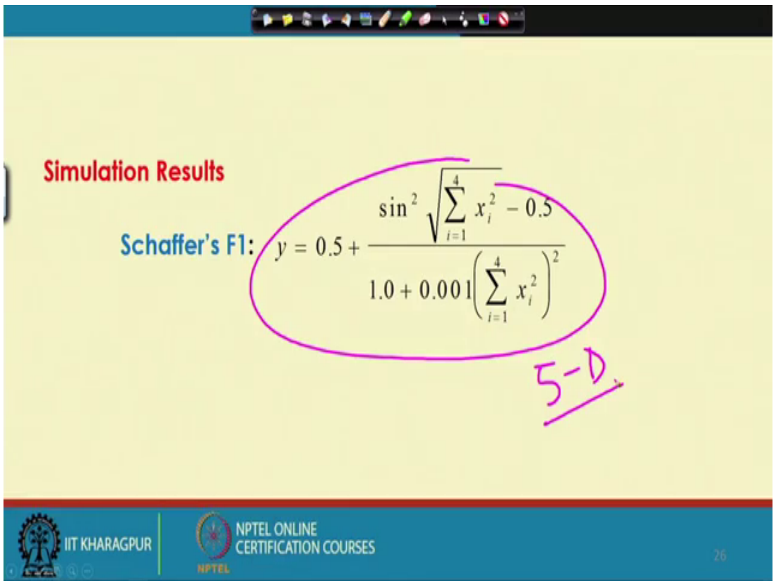
Now, suppose that C 1 is having the minimum Euclidean distance values followed by C 2, C 3 and this is the C N. Now what we do is in lower dimension that is in two dimension considering the numerical value of the Euclidean distance in a higher dimension, we try to draw some circular arc. Now this is the origin in lower dimension and considering the Euclidean distance between the origin of the higher dimension and the connecting weights, which is closest to that origin; we consider first and this particular radius is equal to the Euclidean distance of that particular point in higher dimension.

So, what we do is we draw one circular arc here. So, we draw one circular arc here similarly with the radius equal to the Euclidean distance of the next connecting weight, we draw another circular arc here and we just go on dawing all such circular arcs here. And now on the first circular arc, we can select any point at random.

Now let me consider; let me select this particular point and once I have selected this particular point I also know the Euclidean distance between the closest the connecting weight and the next closest connecting weight. And considering that particular numerical value I draw one circular arc; on the second circular arc and I will be getting one intersection point that is P 2 and following the same principle. So, I can find out P 3; starting from P 2.

Similarly, I can also find out P 4 starting from P 3; so I will be getting all such points here; that means, all capital N points here in 2 d; that means, starting from the higher dimension that is L dimensional space; all capital N data points I can do the mapping to the the lower dimension. Now using this particular the method; there is a self organizing map there is a possibility that we will be getting a very good mapping in the lower dimension.

(Refer Slide Time: 27:28)



Simulation Results

Schaffer's F1: $y = 0.5 + \dfrac{\sin^2 \sqrt{\sum_{i=1}^{4} x_i^2} - 0.5}{1.0 + 0.001\left(\sum_{i=1}^{4} x_i^2\right)^2}$

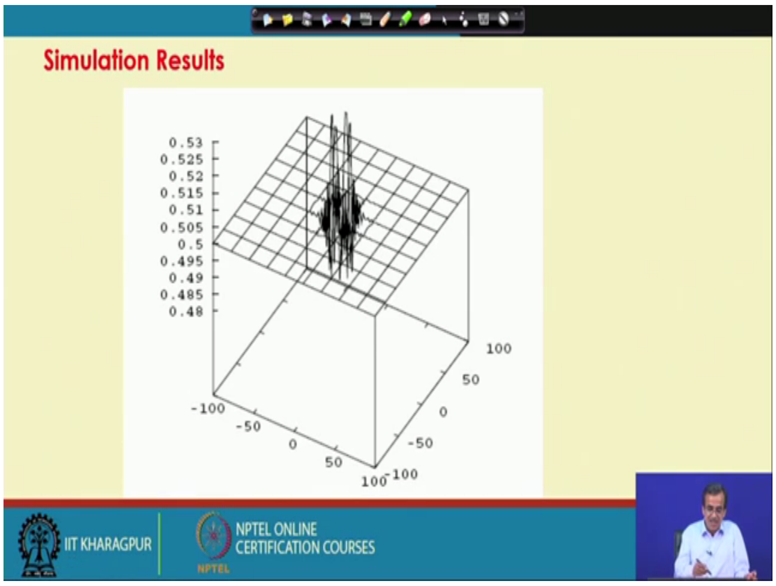
5-D.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Now, to test the performance; so, what you do is we take the help of say one test function say very famous Schaffer's F 1 test function. Now this is the mathematical expression of this particular the Schaffer's test function.

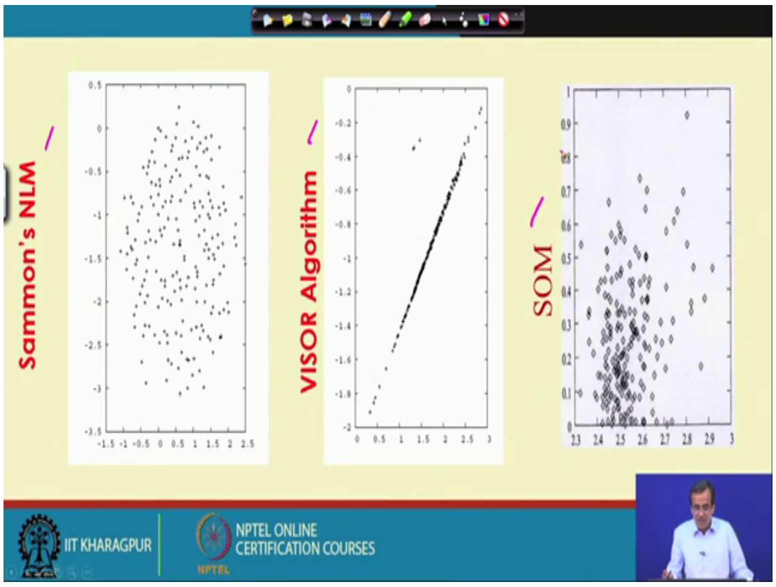
Now, this is in 5 dimension because here we can see that summation i equals to 1 to 4 X i square; that means,. So, this particular function is it 5 D; 5 dimension and we cannot visualize in 5 dimension. So, we can visualize only up to the 3 dimension; now what you do is. So, this particular function cannot be drawn in 5 dimension and that is why approximately actually we draw it in three dimensional like this.

(Refer Slide Time: 28:27)


Simulation Results

So, this is the 3 dimensional plot of this particular test function and here we consider y is a function of only two variables; that means, the other two variables are assumed to be constant and we can find out the approximate plot of this particular the function and our aim is to find out the minimum point of this particular the function.

(Refer Slide Time: 28:54)



Now, what we do is; so we try to compare the quality of the mapped data points using the three different techniques which I have already discussed. For example, so, what we do is supposing that; so this particular test function like Schaffer’s test function we want to minimize with the help of a genetic algorithm.

So, this particular surface of the objective function is in five dimension. So, we cannot visualize; what we do is, we generate 1000 data points on the surface of this particular objective function in 5 dimension. And we do the mapping in two dimension for all 1000 data points using three different techniques like Sammon’s non-linear technique then comes the visor algorithm and the self organizing map.

Now, this shows actually the quality of the mapped data points; now you can see that Sammon’s non-linear mapping. So, here the data points are well distributed and it will help us to visualize; on the other hand if you see the quality of the map data using visor algorithm the data points are coming very close to each other; they are clubbed together and it is bit difficult to visualize them.

And if you see the quality of the map data; obtained using the self organizing map here also the data points are well distributed. So, it is bit easy to visualize; so in terms of visualization capability the Sammon’s non-linear mapping and this the visor is this self organizing map mapping are good compared to the mapping obtained using the visor algorithm.

But, if I compare in terms of the computational complexity, or if you see the visor algorithm is the fastest out of these three. So, within in one iteration only; so it is going to give this particular the map data point. On the other hand the Sammon’s non-linear mapping is an iterative search and it takes a few minutes time to give this particular the mapping.

And if I see the computational complexity of the self organizing map it is slightly less than the Sammon’s non-linear mapping or very close to Sammon’s non-linear mapping. But quality wise if you see; so this particular Sammon’s non-linear mapping and this self organizing map is going to give slightly better data points in lower dimension in terms of the visualization.

That means, in terms of visualization; we can declare that this is the best in terms of computational complexity this is the first test. Now what you do is considering the computational complexity and the quality of the map data, we generally prefer the data points mapped using the self organizing map out of these three; so, this could be the best out of these 3.

Now, we will see how to use this particular information to develop the faster genetic algorithm that is visualized interactive genetic algorithm.

Thank you.