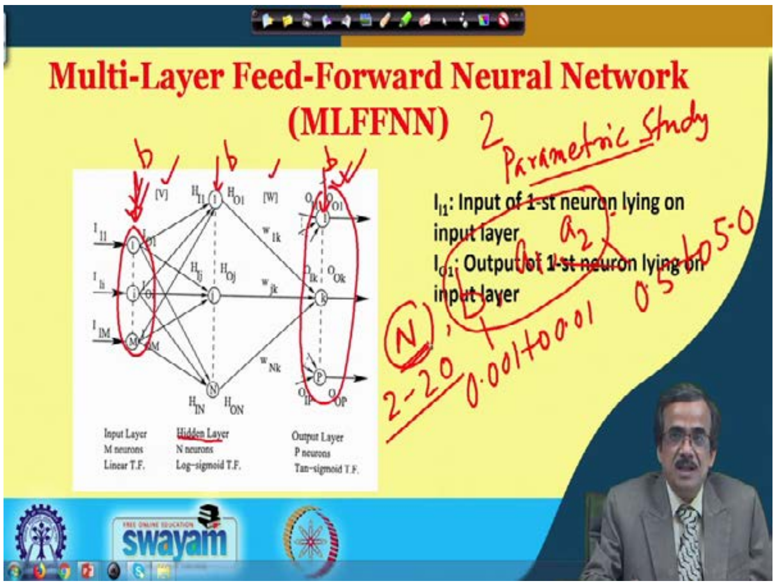
Fuzzy Logic and Neural Networks

Prof. Dilip Kumar Pratihar
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture – 23
Some Examples of Neural Networks (Contd.)

(Refer Slide Time: 00:15)



Multi-Layer Feed-Forward Neural Network (MLFFNN)

I_{I1}: Input of 1-st neuron lying on input layer

I_{O1}: Output of 1-st neuron lying on input layer

We are discussing the working principle of a Multilayered Feed-Forward Neural Network. Now, supposing that I have got a process, whose input-output modeling has to be done, that means, for a known set of inputs what should be the set of outputs that we will have to determine.

Now, the first thing will have to do is, we will have to identify the input parameters and the output variables. Now, these input parameters are to be independent and this would be measurable. Now, here if you once again see the schematic view of this multilayered feed-forward network, we can see that, this is the input layer through which we pass the set of inputs and these inputs are to be in the normalized scale either in the scale of 0 to 1 or from minus 1 to plus 1. Now, how to represent in the scale of 0 to 1 or minus 1 to plus 1; that I have already discussed and here, so this is nothing but the output layer and on the output layer, say we have got say P number of neurons.

Now, this output could be either independent and they could be dependent on each other, also. Now, here you see the connecting weights between the input and the hidden layer is denoted by V and that between the hidden and the output layer is denoted by W. And, we have already mentioned the range for the connecting weights could be either from 0 to 1 or from minus 1 to plus 1.

Now, here, for this particular process, the number of inputs, that is, M is kept fixed and similarly, the number of outputs of this particular process is also known. So, the number of neurons in the input layer, that is nothing but the number of input parameters and the number of neurons on the output layer is nothing but your, the number of outputs. So, the number of neurons in the input layer and the number of neurons in the output layer are known, these are kept fixed.

Now, the architecture of this particular network depends on the number of hidden layers and the number of neurons you put in each of the hidden layers. Now, here, you can see, for simplicity, we have considered only one hidden layer and this hidden layer is having say capital N number of neurons. Now, how to decide the number of hidden layers or how to decide this particular architecture? So, that I am going to discuss a little bit.

Now, before we start that, let me mention that there is no guarantee that we can ensure more accuracy to this particular network by increasing the number of hidden layers. That means, if I use one hidden layer for modeling input-output process and by using the same training set, if I develop another network having say two hidden layers, the network having two hidden layers actually cannot give guarantee that it is going to give more accurate predictions.

Now, how to decide? Now, to decide this actually, what you do is, the first thing we will have to decide that what should be the minimum number of neurons in this particular, the hidden layer. And, let me assume that there is only one hidden layer and the minimum number of neurons to be put in the hidden layer is equal to 2. The reason is, if you do not put the minimum number as 2, we may not be able to capture the nonlinearities of this particular process. So, to capture the nonlinearity of this particular process, the minimum number of neurons to be put in the hidden layer is kept equal to 2.

Now, what should be the optimal number? What should be the optimal number of this particular neurons in this hidden layer? Now, to decide that actually, we will have to carry out one parametric study. Now, before I go for discussing the method of this parametric study, let me once again mention that the performance of this particular network depends on a number of parameters of course, it depends on the connecting weights, that is, V and W.

Now, besides these particular the V and W matrices, the performance of this particular network depends on the number of the neurons, which we are going to put in the hidden layer, that is denoted by capital N, or the number of the neurons, we put on this particular the hidden layer, that is, N and it depends on the bias value. Now, for simplicity, we have not shown the bias value, but it depends on the bias value. Now, if you want to put the bias value, so we will have to put that particular the bias value. So, I can put this particular the bias value, here.

It depends on the coefficient of transfer function, that is, a_1 that means, your this log sigmoid transfer function. It also depends on the coefficient of transfer function of the output layer, that is nothing but a_2, and so on. So, these are the parameters for which we will have to determine the set of optimal values.

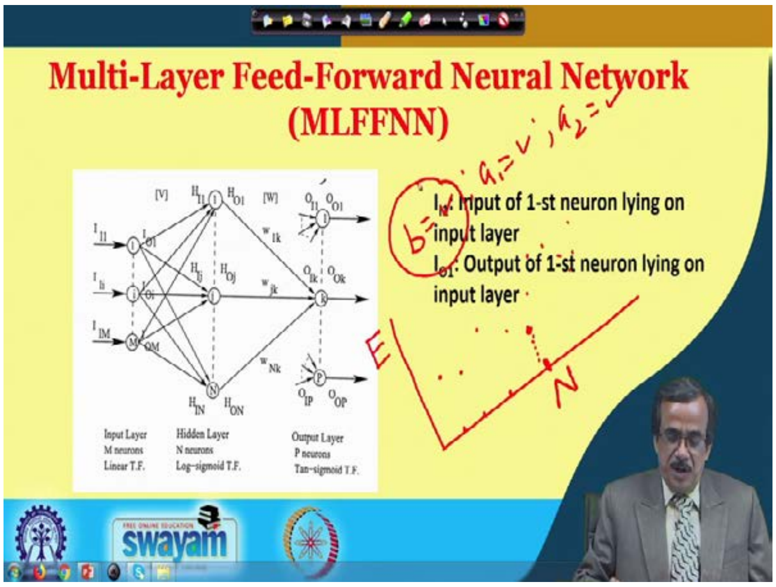
Now, to carry out this parametric study actually, what you do is, we first try to fix the range of each of these particular parameters. For example, to fix the range, the minimum number is kept equal to 2. Now, supposing that I am just going to put the range for N we say 2 to say 20. Now, I can also put the range for this particular b. So, might be it is 0.001 to say 0.01 something like this and for this a_1 and a_2, I can also put the range, so might be it is from 0.5 to say 5.0, and so on.

Now, once you have decided these particular ranges for the parameters, now we are in a position to start the parametric study and the purpose of parametric study is to decide, what should be the topology of this particular network. And, what should be the parameters, the other parameters and let me repeat that this connecting weight values, those are generated in the normalized scale using the random number generator.

Now, let us see, how to carry out this particular the parametric study. Now, the first thing we will have to do is, we will have to vary the number of the hidden neurons starting from 2 to 20 say and regarding the other parameters like your b, a_1 and a_2. So, we try to find out their mid values and these parameters are set to their corresponding mid-values and I am just going to vary the number of neurons to be present in the hidden layer.

Now, if I carry out this particular study so there is a possibility that I will be getting.

(Refer Slide Time: 08:39)



Multi-Layer Feed-Forward Neural Network (MLFFNN)

I_{1I}: Input of 1-st neuron lying on input layer

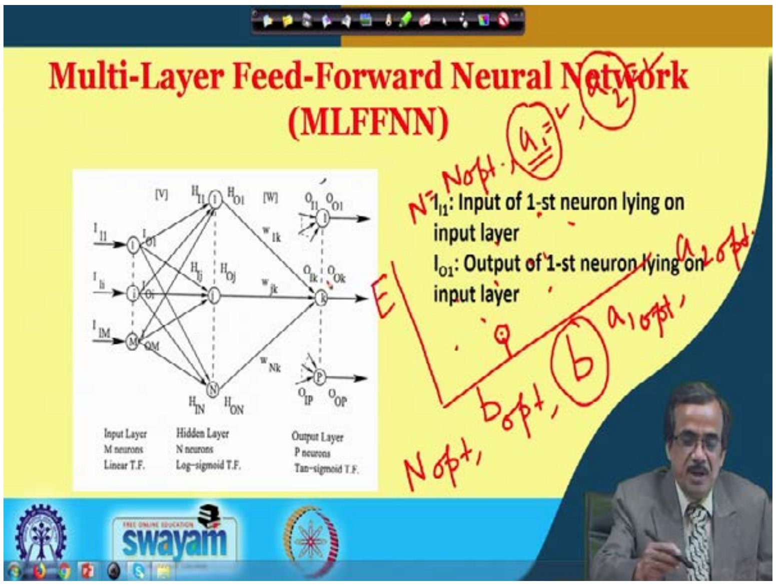
I_{O1}: Output of 1-st neuron lying on input layer

So, with the results of this study, what you can do is, you can plot say the error of this particular network, that is error in prediction, and here, we have got the number of neurons to be put in the hidden layer and we keep some fixed value for say b, we keep some fixed values for a_1 and a_2.

Now, what we do is, we start with the two number of hidden neurons. So, say this is going to indicate the number of hidden neurons, so I will be getting some error. So, error could be here, then I use like three number of hidden neurons, might be the error could be here, then I use the four number of hidden neurons might be the error could be here. So, what we do is, we try to find out the nature of the performance of this particular network.

And, then, we try to select that number of hidden neurons, which gives actually the optimal performance in terms of error in prediction. Now, supposing that, say this is the number of neurons corresponding to which I am getting the optimal performance. Now, what you do is, the value of N is kept fixed to this particular value and next, we go for the variation of this particular b.

(Refer Slide Time: 10:06)



Multi-Layer Feed-Forward Neural Network (MLFFNN)

I_{I1}: Input of 1-st neuron lying on input layer

I_{O1}: Output of 1-st neuron lying on input layer

Now, what we do is, so exactly in the same way, we carry out the study. So, now, here there will be error and here there will be the bias value and N is kept fixed to the optimal value, which you have already determined in the previous step. And, a_1 is kept fixed with mid value, a_2 is kept fixed with mid-value and once again, for different values of b, so we try to find out the error and we try to locate that value of b corresponding to which, we are getting the minimum error. So, we select the near-optimal value for this particular the b.

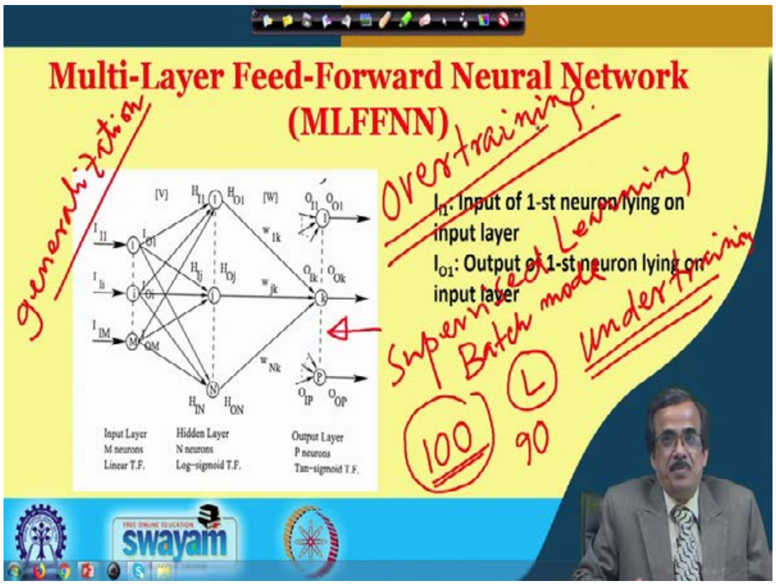
Next we start this particular study by wearing this particular a_1 and we try to find out what should be the optimal or the near optimal value for this particular a_1. Next, we start, this particular study actually with the variation of this particular a_2 and we try to find out what should be the near optimal value for this particular the a_2.

So, by carrying out this particular study, so we can find out what should be the near optimal value for this particular the number of hidden neurons, what should be near optimal value for this particular the bias value, then comes what should be the near optimal value for this a_1 and the near optimal value for this particular a_2. And once you have got this particular thing, now we are in a position to decide at least approximately, what should be the topology of this particular network and what should be the other parameters.

Now, here I just want to make a comment, this process of carrying out the parametric study is an approximate one, because here, the performance of this network depends on all the parameters simultaneously. But, what we did is, we consider one parameter at a time that means, this is an approximate way of determining the near optimal parameters of this particular the network. So, this is the way actually, we can approximately determine like, what should be the topology and what should be the other parameters for this particular the network.

Now, as I discussed in my last lecture that this particular network has to be trained and if I go for the supervised learning, this supervised learning actually can be implemented either in the incremental mode of training or in the batch mode of training.

(Refer Slide Time: 12:47)



Multi-Layer Feed-Forward Neural Network (MLFFNN)

I_{I1}: Input of 1-st neuron lying on input layer

I_{O1}: Output of 1-st neuron lying on input layer

Now, I am not going to repeat because this I have already discussed in much more details that the principles of the incremental training, the batch mode of training, their relative merits and demerits, these things we have already discussed. But, now, I am just going to concentrate a little bit on the batch mode of training for this supervised learning.

Now, supposing that in batch mode of training, say I have got a large number of training scenario say capital L number of training scenarios. And, as I mention several time, out of these capital L, the first one I am just going to pass and depending on the connecting weights, the coefficient of transfer function and other things, so I will be getting the set of outputs.

Now, this set of calculated outputs will be compared with the respective target values just to find out the error. And, once you have got error at each of this particular output neurons, I can find out what should be the combined error and that corresponds to a particular training scenario. Then, we take the help of the second training scenario, I will be getting another, the total error here.

Then, the third one we put their and I just go for the L-th training scenario and once you have passed all these L training scenarios, I will be able to find out, what is the total error, what is the average error and based on this particular, I will have to propagate back, so that I can modify the different variables and ultimately through a large number of iterations, I get that particular optimal or the near-optimal network.

Now, here I just want to make a point like, if I just go for the batch mode of training, we will have to be careful that we have used sufficient number of training scenarios. Now, supposing that, this particular network is having say approximately, say 100 design variables. Now, if it is having 100 design variables or your parameters, which are to be optimized, whose modified values are to be determined. So, what will have to do is I will have to pass at least 100 training scenarios. So, either 100 or slightly more than 100 training scenarios I will have to pass, while carrying out the batch mode of training.

Now, let us see, what happens if I just pass say less number of training scenario, this I have already mentioned and let me try to repeat once again. Now, supposing that I have got 100 design variables. Now, if I pass say 90 training scenarios that is a case of the under-training of these particular the network. That means, mathematically, if you want to put, supposing that I have got 10 unknowns and I have got say 9 equations. So, if the 10 unknowns and nine equations we cannot solve mathematically the same thing is to here, if I have got less number of training scenarios compared to the number of unknowns, this will be a clear example of your under-training.

And, if there is under-training for this particular network and if I pass a set of inputs, will be getting some output for this particular output cannot be actually believed and this network actually does not know anything of the physical problem. So, if you just pass say one set of training scenarios, we are going to get the outputs for a set of inputs. But, this particular input-output relationship may not be reliable.

Now, on the other hand, if I use a very large number of training scenarios and if I train this particular network through a large number of iterations, now let us see what happens. Now, if I just train this particular network for a large number of iterations using a very large number of training scenario, supposing that I have got only 100 design variable and if I use 1000 training scenario and train this particular network for a very large number of iterations, say 5000 iterations. What will happened to this particular the network? Will this network be going to give a very good performance? The answer is no.

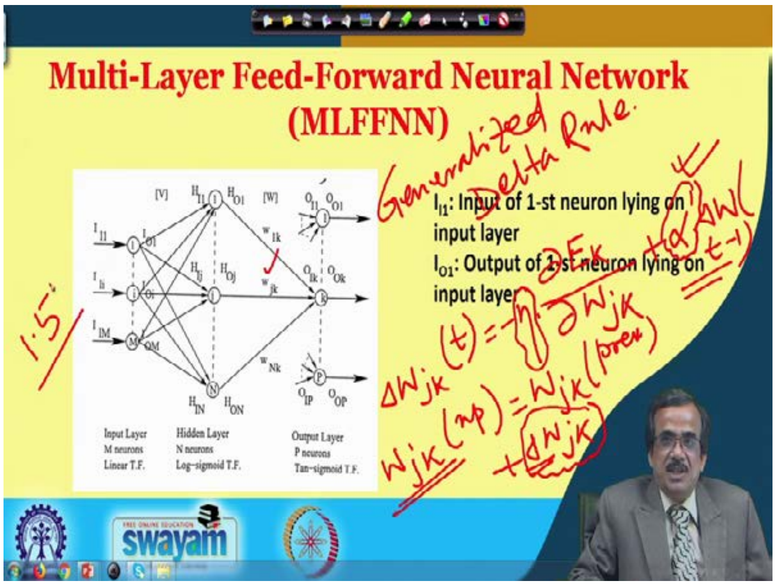
Now, here, there will be some sort of overtraining. So, this overtraining is also actually is not good for this particular network and the reason, I am just going to discuss in details. Now, if there is overtraining, this particular network will try to remember actually the training scenarios and what will happen; the moment we pass the unknown test scenarios, it will try to remember only the training scenario and it may not perform in the optimal sense for the unknown test scenarios, because its adaptability will be lost or its generalization capability will be lost, if I just go for the overtraining.

Now, as I told this particular network should have a very good generalization capability. So, this generalization capability is actually a desired property for this particular network and if you want to ensure; so what will have to do is. So, we should not go for under-training and we should not go for overtraining, and in other words, actually we should go for a perfect training for this particular network.

Now, supposing that so we have selected the number of training scenarios required for carrying out the batch mode of training and while carrying out this particular batch mode of training, there is a possibility that we will be a facing another problem now, that problem actually I am just going to explain which I have learnt from my own practical experience. So, this, I am going to discuss.

Now, supposing that I am carrying out the batch mode of training for this supervised learning and I am using the suitable number of training scenarios, and I am using the generalized delta rule, the back propagation algorithm that is nothing but supposing that I want to update this particular the connecting weight.

(Refer Slide Time: 20:33)



Multi-Layer Feed-Forward Neural Network (MLFFNN)

I_{I1}: Input of 1-st neuron lying on input layer

I_{O1}: Output of 1-st neuron lying on input layer

Generalized Delta Rule.

$$\Delta w_{jk}(t) = -\eta \frac{\partial E_k}{\partial w_{jk}} + \alpha' \Delta w(t-1)$$

$$w_{jk}(up) = w_{jk}(prev) + \Delta w_{jk}$$

So, what I will have do is, so $\Delta w_{jk} = -\eta \frac{\partial E_k}{\partial w_{jk}} + \alpha' \Delta w(t-1)$. This is actually the generalized delta rule. This I have already discussed.

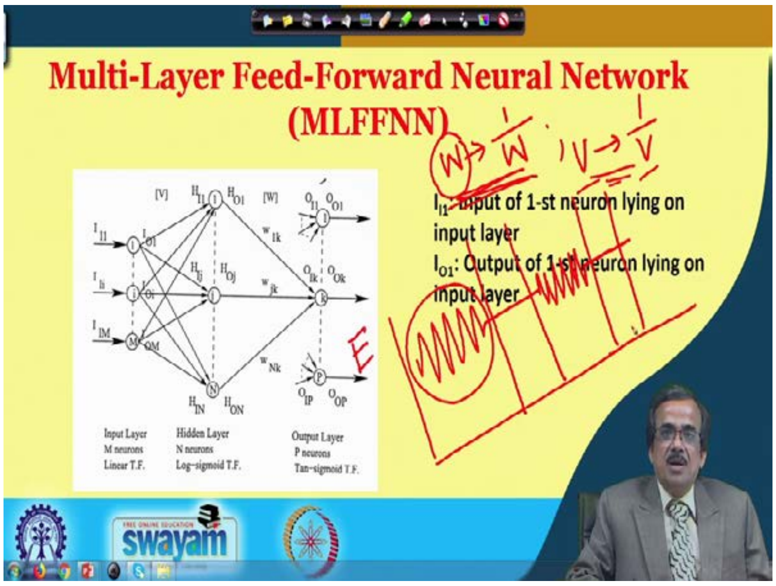
Now, using this generalized delta rule, I will have to update this connecting weights. And, I have already discussed that η is nothing but the learning rate and it has got the range of 0 to 1, then α' is nothing but the momentum constant, and the purpose of using the momentum constant I have already discussed. If you want to provide some sort of damping effect to the network and if you want to provide some sort of cushioning effect to this particular network, we will have to use this particular your momentum term.

Now, here there is one mathematical reason, why do you put this particular your the momentum term. Now, supposing that the way we updated this particular network is nothing but $W_{jk}(updated) = W_{jk}(previous) + \Delta W_{jk}$. So, this is the way actually we update. Now, supposing that this particular delta W value is found to be positive for a large number of consecutive iterations.

Now, this ΔW it could be either positive or negative, but fortunately or unfortunately say I am getting the positive values for a large number of consecutive iterations. Then, what will happen to this particular your ΔW_{jk} updated?. Now, there is a possibility that this particular ΔW_{jk} will come out of the range that means, it will become say greater than 1, supposing the this is becoming w is becoming say 1.5, whereas the maximum limit for this W is 1. Now, what will happen to this particular network?

Now, as I mention several times that network does not know anything of the physical problem. Now, even if the W has come out of the range or the V has come out of the range, if you pass the set of inputs, the network is going to give you the output, but that particular output may not be reliable, because these particular connecting weight values have come out of the range. And, if this is the situation, that particular situation is actually known as actually the stability problem of this particular network. That means, the network has become unstable, it has lost its balance and so this particular network is going to behave in a very the erratic way and you may not get a very good prediction for a set of inputs.

(Refer Slide Time: 24:55)



Multi-Layer Feed-Forward Neural Network (MLFFNN)

$W \rightarrow \frac{1}{W}$ $V \rightarrow \frac{1}{V}$

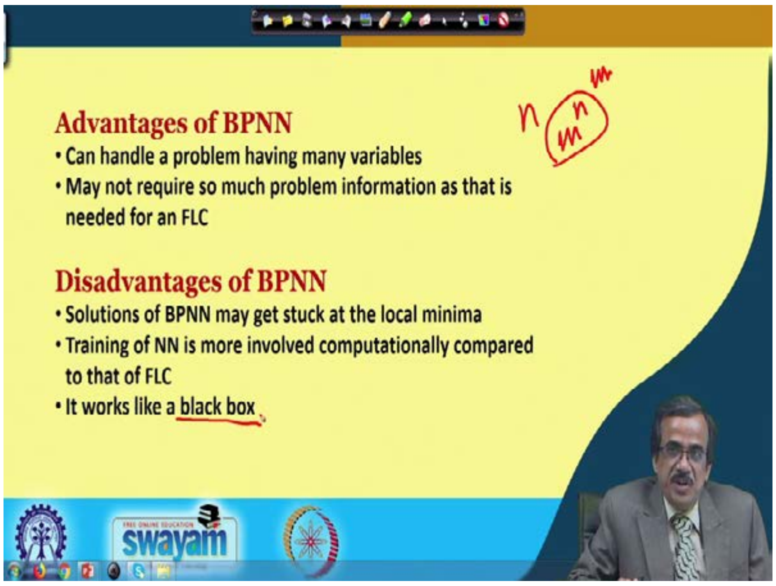
I_{I1}: Input of 1-st neuron lying on input layer

I_{O1}: Output of 1-st neuron lying on input layer

Now, here, we can put one remedy, the remedy is something like this. So, if W or V becomes greater than 1, so it has become out of range. So, what you do is, in place of W what you put is, 1 divided by W or in place of V, you put like 1 divided by V. So, by doing that actually, what you can do is, W becomes greater than 1. Now, 1 divided by W will be less than 1, the same is true for your V and 1 by V and by doing that, once again forcefully we are going to put that particular limit within the range and what will happen is, your network once again will try to become stable.

But, if you see this particular performance like if I just say this is error and what will happen the moment you put this particular correction, there will be a sharp change of this particular performance and after that, it will reach once again the stable region. And, once again, if it becomes out of range like V and W and if you put this particular correction, so once again, it will have this type of erratic behavior and then, it will try to reach that particular stable zone. Now, within this particular erratic behavior zone, we cannot believe the network, and we will have to believe in this particular range and if you want to use this network for making some predictions.

(Refer Slide Time: 26:35)



Advantages of BPNN

- Can handle a problem having many variables
- May not require so much problem information as that is needed for an FLC

Disadvantages of BPNN

- Solutions of BPNN may get stuck at the local minima
- Training of NN is more involved computationally compared to that of FLC
- It works like a black box

$n \quad m^n \quad m^m$

So, these are all actually the facts, which we should understand, if we want to design a very efficient multilayer feed forward network. Now, I am just going to quickly look into the merits and demerits of this multilayer feed forward network. Now, this particular multilayer feed forward network can handle a large number of variables.

Now, if you remember while discussing the fuzzy reasoning tool, we have already discussed that if the number of input parameters increase and handling or modeling that particular process using fuzzy reasoning tool will be more difficult, because the number of rules is going to increase exponentially. For example, say this I have already discussed, if a particular process is having say n number of input parameters and to represent each input parameter, if I use m linguistic terms in fuzzy reasoning tool. So, what will happen? The total number of rules become m^n, it is a very high number large number and this is known as the curse of dimensionality.

Now, here for the same problem if I use the multi-layered feed forward network, so we will be getting efficient modeling. So, in place of this fuzzy reasoning tool we can use this multilayered feed forward network. The next is, it may not require so much problem information, which you need in fuzzy reasoning tool or fuzzy logic controller because while discussing the fuzzy reasoning tool, we have already discussed that its performance depends on the database and the rule base.

And, if the designer wants to design initially the database and the rule base of the fuzzy reasoning tool, he or she should know or he or she should have at least some initial information of the process to be controlled. But, here, in neural network training, we may not need so much problem information. Now, based on the experiments, if you can collect some input-output relationships that you can directly used to train the network and once it is trained the network will be able to predict the input output relationships in a very efficient way.

Now, disadvantages of the multilayer feed forward network or demerits. The solution of this back propagation neural network that is that multilayered feed forward network trained using back propagation algorithm, the solution may get stuck at the local minima. So, this I have already discussed, because initially we do not know what should be the nature of the training nature of the error surface during the training. Now, if the error surface is found to be unimodal. So, this back propagation algorithm, which is nothing but the steepest decent algorithm is going to hit the globally minimum solution very easily.

But, supposing that the nature of the error surface is such, it is having multiple modes there are many such ups and downs, so in that case there is a possibility, the solution of this back propagation neural network or multilayered feed forward network trained using back propagation algorithm may get stuck at the local minima and it may not be able to reach the globally minimum solution.

Now, if you see the computational complexity, the training complexity of a neural network and the training complexity of a fuzzy reasoning tool, the training complexity of the multilayered feed forward network is more compared to the computational complexity of the fuzzy reasoning tool. So, the training of neural network is computationally more complex compared to that of the fuzzy reasoning tool.

And, there is another drawback, I should say that is your this particular multilayered feed forward network is nothing but a black box. For example, say we have got a set of training scenarios and we train a particular the network. Now, for training, what you need is for a set of training input parameters, what should be the set of output parameters and if those relationships are known for a large number of training scenarios, we can train the network, but during the training what is happening inside the network, we, the designers, do not know. But, ultimately through a large number of iterations, it is going to give rise to one optimal or the near optimal network. So, this particular multilayered feed forward network works like a black box.

Thank you.