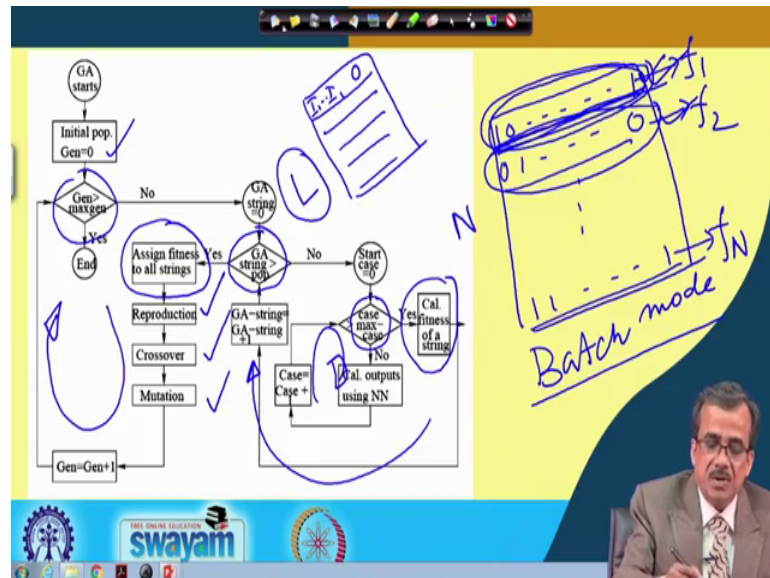


Fuzzy Logic and Neural Networks
Prof. Dilip Kumar Pratihari
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture – 32
Optimal Designs of Neural Networks (Contd.)

(Refer Slide Time: 00:15)



Now, we have discussed how to encode the number of hidden layers, the number of neurons to be present in the each of the hidden layers that is the topology or the architecture of the network and how to encode the connecting weights the coefficient of transfer function the bias values inside that particular the GA string. So, ultimately the GA string is going to carry information the complete information of the neural network; that means, it will carry the information related to topology, it will carry the information related to the connecting weights, then the information related to the bias value the coefficient of transfer function.

So, this particular GA string is going to carry the whole information of this particular the network and we are going to take the help of the batch mode of training to optimize or to evolve this particular the network. Now, this shows actually one flow chart or the schematic view like how does it work. Now, let me explain with the help of this particular flowchart the working principle of this genetic neural system.

Now, as I told that genetic algorithm is nothing, but a population based approach. So, I have got a population size N and these particular population of solutions are generated at random. Supposing that the first solution is something like this these are binary coded GA the second solution is something like these and the last solution is something like this. Now, if I concentrate on a particular the GA string that is the first GA string, it will carry the full information or the whole information of these particular the network.

Now, let us see how to implement the batch mode of training for this type of the network. So, I am just going to discuss the batch mode of training of this particular the network. Now, GA starts with a population of solution and we create the initial population at random we put generation is equals to 0. Now, here we have got a check the generation whether it is greater than equals the maximum number of generations. Now, if it is yes that is the end of the algorithm and if it is no so, we concentrate on the first GA string; that means, we are going to concentrate on the first GA string and we put GA string equals to 0.

And, here there is another check whether the GA string is greater than the population size. Now, if it is no then we start with the training case there is the first training case; supposing that we have got some training scenarios and the total number of training scenarios let me consider I have got capital L number of training scenarios. Now, a particular training scenario carries information of the input and your the output similarly we have got capital L number of training scenarios. Now, corresponding to the first GA string, so, we are going to concentrate on the training cases or the training scenarios.

Now, here so, corresponding to the first GA string my neurotic is ready and I am passing all the training scenarios one after another. For example, say here I have got a check where the training scenario or the case is greater than the maximum case that is a maximum number of training scenario. Now, if it is no then we calculate the output of the neural network. So, as I told that this particular neural network is indicated by. So, this particular the GA string.

So, we will be getting the output of this particular network and we use case equals to case plus 1; that means, I am just going to pass all the training scenarios one after another. The moment it satisfies so, this particular condition, so, what we do is we calculate the fitness of the your the GA string; that means, after passing all the training

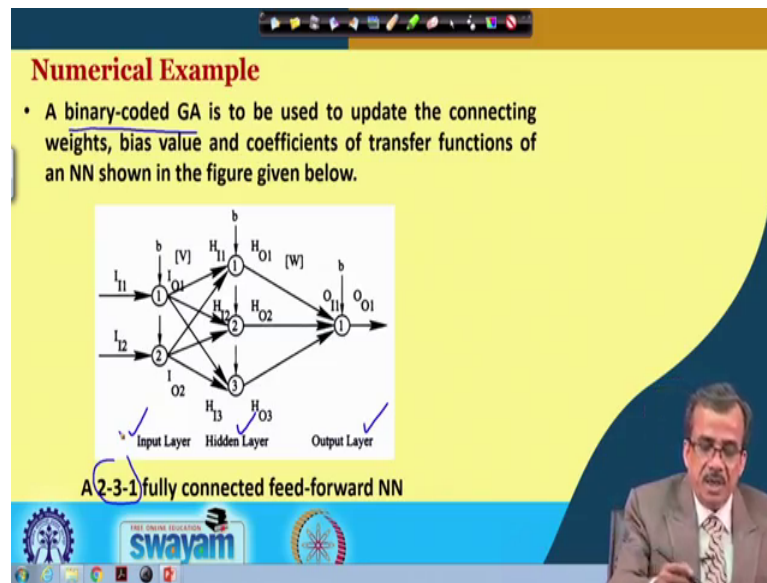
scenarios all capital L training scenarios we try to consider the fitness we try to calculate the fitness of these particular the GA string and supposing that the fitness of the GA string is denoted by f_1 and so, here we have got GA string equals to GA string plus 1; that means, we go for the second GA string; that means, we are going to concentrate on these particular your the second GA string.

And, once again for this particular your the second GA string. So, my network is ready once again we will pass all the training scenarios and we will be getting so, these particular f_2 and this particular process will go on and go on and the moment it reaches this particular the criteria that is GA string is greater than the population size; that means, your the fitness information for the whole population is ready for us; that means, we have got the fitness information that is f_1, f_2 up to your f_n .

And, once you have got the fitness information for the whole population, now we are going to modify so this particular the population of solutions using the operators like reproduction crossover and mutation. Now, the principle of reproduction crossover and mutation; so, those things actually I have discussed in some of the earlier lecture. Now, so, this particular process will go on and go on and this will complete actually one iteration or one generation of this particular GA and GA through a large number of iteration we will try to find out your some optimal design of these particular the networks.

And, as I told that there is a possibility that you will be getting multiple solutions; that means, the multiple optimal neural networks and you can use any one out of these multiple optimal neural networks for predicting the input output relationship. Now, this is the way actually the genetic neural system works the working principle of this genetic algorithm genetic neural system we have already discussed.

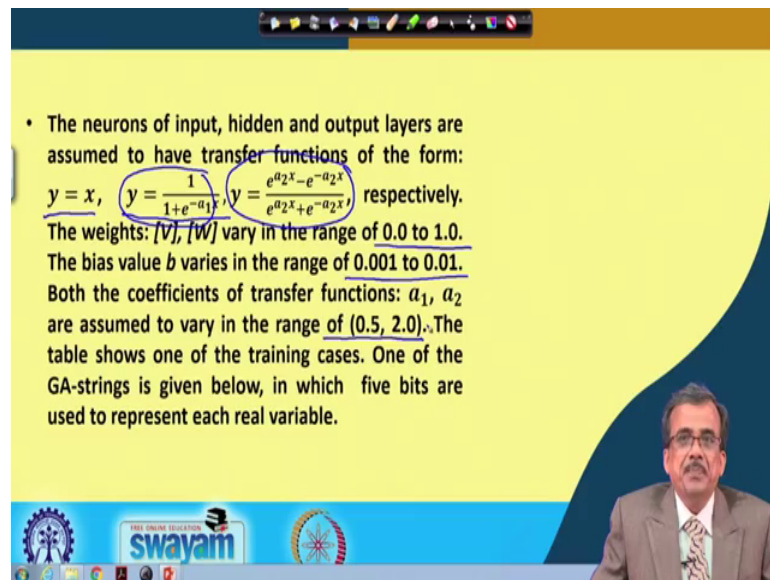
(Refer Slide Time: 08:01)



And, now actually what we are going to do; we are going to solve one numerical example just to make it more clear. Now, here I am just going to solve one numerical example and we are going to solve this numerical example and I am going to give the statement of this numerical example.

A binary coded genetic algorithm is used to update the connecting weights, coefficient of transfer function of a neural network as shown below. So, this is actually a very simple network having say three layers – input layer, hidden layer and output layer. So, on the input layer there are two neurons; on the hidden layer there are three neurons and the output there is one neuron. So, this is nothing, but actually a 2-3-1 fully connected the network and let us see how to optimize so, this particular your network with the help of a binary coded algorithm.

(Refer Slide Time: 09:14)



• The neurons of input, hidden and output layers are assumed to have transfer functions of the form:

$y = x$, $y = \frac{1}{1+e^{-a_1x}}$, $y = \frac{e^{a_2x} - e^{-a_2x}}{e^{a_2x} + e^{-a_2x}}$ respectively.

The weights: $[V]$, $[W]$ vary in the range of 0.0 to 1.0.

The bias value b varies in the range of 0.001 to 0.01.

Both the coefficients of transfer functions: a_1 , a_2 are assumed to vary in the range of (0.5, 2.0). The table shows one of the training cases. One of the GA-strings is given below, in which five bits are used to represent each real variable.

Now, the rest of the statement of the problem is as follows. The neurons of the input and hidden layer and output layers are assumed to have transfer function of the form y equals to x , that is the linear transfer function for the input layer, y equals to one divided by one plus e raised to the power minus $a_1 x$ that is nothing, but is your the log sigmoid transfer function for the hidden layer. And y equals to e raised to the power $a_2 x$ minus e raised to the power minus $a_2 x$ divided by e raised to the power $a_2 x$ plus e raised to the power minus $a_2 x$. So, this is actually the tan sigmoid transfer function.

And, the connecting weights v and w we are going to vary in the range of 0 to 1, the bias value will vary in the range of 0.001 to 0.01 and the coefficient of transfer function that is a_1 , a_2 are going to vary in the range of 0.5 to say 2.0. Now, we are going to show one training scenario and for this particular training scenario actually we will have to find out what should be the output and that output we will have to compare with the target output just to find out the deviation in prediction.

(Refer Slide Time: 10:53)

Table: Training cases


Sl. No.	I_1	I_2	O
1	0.6	0.7	0.9
.	.	.	.
.	.	.	.
.	.	.	.
L	.	.	.

2-3-1

10110 01101 10001 01011 11011 00011 11001 11110 11101 10101 01110 11000

v_{11} v_{12} v_{13} v_{21} v_{22} v_{23} w_{11} w_{21} w_{31} a_1 a_2 b

- Determine the deviation in prediction for the training case shown in the above table, corresponding to the above GA-string.



Now, the training scenario as I told is something like this and the training scenario is nothing, but the input output relationship now there are L capital L obtaining scenarios the out of that the first one is as follows. If I_1 is 0.6 and I_2 is 0.7, then output is 0.9. So, this is nothing, but is your the input-output relationship, known input-output relationship.

Now, if you concentrate on the GA string the GA string will carry information of this particular network; that means, your these 2-3-1 neural network and it is having the fixed architecture and we are going to use 1 2 3 4 5 5 bits to represent each of these particular the design variables. For example, 5 bits are used to represent v_{11} ; the next 5 bits are used to represent v_{12} and so on. So, this particular GA string is going to carry the full information or the whole information of a fixed architecture that is 2-3-1 neural network. And, as I told that our aim is to determine the deviation in prediction so, for this particular the training scenario.

Now, let us see how to find out so that particular your the deviation in prediction

(Refer Slide Time: 12:34)

Solution:

- The variable v_{11} is represented by the binary sub-string 10110. Its decoded value is found to be equal to 22. It varies in the range of (0.0,1.0). Using the linear mapping rule, its real value can be determined as follows:

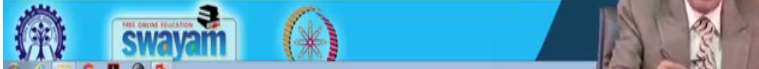
$$v_{11} = 0.0 + \frac{1.0 - 0.0}{2^5 - 1} \times 22 = 0.709677$$

Linear mapping

$$v_{11} = v_{11}^{min} + \frac{v_{11}^{max} - v_{11}^{min}}{2^l - 1} \times DV$$

$$v_{11} = 0.0 + \frac{1.0 - 0.0}{2^5 - 1} \times 22$$

- Similarly, the real values of all variables represented by the above GA-string can be calculated and those are shown in the following table.



Now, to determine the deviation in prediction, so what you will have to do is the first thing you will have to find out the corresponding to this particular GA string. So, you will have to find out the decoded value.

(Refer Slide Time: 12:53)

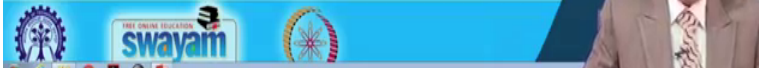
Table: Training cases

Sl. No.	I_1	I_2	O
1	0.6	0.7	0.9
.	.	.	.
.	.	.	.
L	-	-	-

$$D.V. = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 16 + 4 + 2 = 22$$

$$\underbrace{10110}_{v_{11}} \underbrace{01101}_{v_{12}} \underbrace{10001}_{v_{13}} \underbrace{01011}_{v_{21}} \underbrace{11011}_{v_{22}} \underbrace{00011}_{v_{23}} \underbrace{11001}_{w_{11}} \underbrace{11110}_{w_{21}} \underbrace{11101}_{w_{31}} \underbrace{10101}_{a_1} \underbrace{01110}_{a_2} \underbrace{11000}_{b}$$

- Determine the deviation in prediction for the training case shown in the above table, corresponding to the above GA-string.



For example, say if I just want to find out the decoded value corresponding to this particular your the 5 bits which are used to represent the v_{11} . So, this is nothing, but 10110. The place values are as follows 2 raised to the power 0, 2 raised to the power 1, 2 raised to the power 2, 2 raised to the power 3, 2 raised to the power 4 and the decoded

value is nothing, but 1 multiplied by 2 raised to the power 4 plus 1 multiplied by 2 raised to the power 2 plus 1 multiplied by 2 raised to the power 1. Now, this is nothing, but is your one 16 this is nothing, but is your 4 and this is 2. So, this is nothing, but is your 22.

So, the decoded value corresponding to these 5 bits used to represent v_{11} is nothing, but is your 22 and once you have got the decoded value, now we can find out the linear mapping rule. We can use a linear mapping rule to find out the real value corresponding to that binary sub string. Now, what you do is now we have already discussed the decoded value of that and we know the range the range is nothing, but 0.0 to 1.0.

So, using the linear mapping rule which I have already discussed the linear mapping rule so, we can find out the real value corresponding to this v_{11} . So, v_{11} is nothing, but v_{11} minimum that is your 0.0 plus v_{11} maximum that is your 1.0 minus v_{11} minimum that is nothing, but is your 0.0 divided by 2 raised to the power l ; l is nothing, but your number of bits used to represent that is 2 raised to the power 5 minus 1 multiplied by the decoded value.

Now, if I just write down this rule that is v_{11} is nothing, but v_{11} minimum plus your v_{11} maximum minus v_{11} minimum 2 raised to the power l minus 1 multiplied by the decoded value. So, this is nothing, but the linear mapping rule. Now, here small l is nothing, but is your 5 because we are using 5 bits to represent v_{11} . So, very easily we can substitute all the numerical values and we can find out what should be the real value corresponding to your v_{11} .

(Refer Slide Time: 15:59)

Table: Real values of the variables represented by a GA-string

Sl. No.	Variable	Binary String	Decoded value	Range	Real value
1	v_{11}	10110	22	0.0, 1.0	0.709677
2	v_{12}	01101	13	0.0, 1.0	0.419355
3	v_{13}	10001	17	0.0, 1.0	0.548387
4	v_{21}	01011	11	0.0, 1.0	0.354839
5	v_{22}	11011	27	0.0, 1.0	0.870968
6	v_{23}	00011	03	0.0, 1.0	0.096774
7	w_{11}	11001	25	0.0, 1.0	0.806452
8	w_{21}	11110	30	0.0, 1.0	0.967742
9	w_{31}	11101	29	0.0, 1.0	0.935484
10	a_1	10101	21	0.5, 2.0	1.516129
11	a_2	01110	14	0.5, 2.0	1.177419
12	b	11000	24	0.001, 0.01	0.007968

Now, the same principle actually we can use to find out the decoded value and the real value for each of the variables. Now, for this particular v_{11} , so, I have already discussed like how to find out actually the real value. now, you follow the same principle to find out the real values for each of these particular connecting weights v_{11} then 12, 13, v_{21} , v_{22} , v_{23} , w_{11} , w_{21} and w_{31} and we can find out their corresponding real values. So, we can find out their corresponding real values,.

Now, the range for your this particular a_1 that is the coefficient of transfer function for your the log sigmoid transfer function. Now, here the range is 0.5 to 2.0 and once again by following the same principle you can find out what should be the real value. Similarly corresponding to a_2 , I can find out the real value for b , I can find out the real value and the range for b is nothing, but 0.001 to 0.01.

So, I can find out the real values for each of these particular your the variables and a once you have got the real values so, my network is ready and once this particular network is made ready now, I can pass actually your the training scenario that is your the known input output relationship.

(Refer Slide Time: 17:38)

- The outputs of the neurons of input layer are calculated as follows:
$$I_{01} = I_{I1} + b = 0.6 + 0.007968 = 0.607968$$
- The inputs of different neurons of hidden layer after adding the bias value is turning out to be like the following:
$$I_{02} = I_{I2} + b = 0.7 + 0.007968 = 0.707968$$

For example, if I pass the set of inputs I will be getting the output, let us see how does it work. It is very simple. So, this I O1 that is nothing, but the output of the first neuron lying on the input layer is nothing, but is your I I1 that is the input of the first neuron lying on the input layer plus b is the bias value and if you just substitute the numerical values, so, you will be getting this I O1.

Similarly, I will be getting I O2 that is the output of the second neuron lying on the input layer is nothing, but the input of the second neuron lying on the input layer plus bias value and if you substitute the numerical values. So, it will be getting this as I O2.

(Refer Slide Time: 18:51)

$$H_{11} + b = I_{01} \times v_{11} + I_{02} \times v_{21} + 0.007968 = 0.690644$$
$$H_{12} + b = I_{01} \times v_{12} + I_{02} \times v_{22} + 0.007968 = 0.879539$$
$$H_{13} + b = I_{01} \times v_{13} + I_{02} \times v_{23} + 0.007968 = 0.409883$$

- The outputs of hidden neurons are found to be as follows:

And, once you have got this particular thing so, now, actually very easily I can find out so, what should be the your the input of your the hidden neurons.

So, this your H_{11} that is the input of the first neuron lying on the hidden layer and I am also adding some bias value b . So, I will be getting. So, this H_{11} is nothing, but I_{01} multiplied by v_{11} plus I_{02} multiplied by v_{21} plus the bias value is nothing, but these, so, I will be getting this particular your the input. Now, similarly I can find out the input of the neuron second neuron lying on the hidden layer plus bias value. I can also find out the input of the third neuron lying on the hidden layer plus bias value and once you have got this particular the inputs of the hidden neurons then using that the transfer function so, very easily I can find out what should be the output.

(Refer Slide Time: 19:57)

$H_{01} = 0.740219$

$H_{02} = 0.791418$ ✓

$H_{03} = 0.650545$

The input of the neuron lying on output layer after adding the bias value is turning out to be like the following:

$$O_{I1} + b = H_{01} \times w_{11} + H_{02} \times w_{21} + H_{03} \times w_{31} + b$$
$$= 1.971413$$

The slide also features a Swayam logo and a presenter in the bottom right corner.

Now, so, this H_{01} is nothing, but the output of the first neuron lying on the hidden layer then comes here H_{02} is the output of the second neuron lying in the hidden layer and that is coming to be equal to 0.791418. Similarly this H_{03} that is the output of the third neuron lying on the hidden layer and this is coming equal to be your 0.65 then 0545 and once you have got your the output of your the hidden neurons so, very easily we can find out what should be the input of the neuron lying on the output layer.

So, this O_{I1} is nothing, but the input of the first neuron lying on the output layer and we are adding the bias value and this O_{I1} is nothing, but H_{01} multiplied by w_{11} , H_{02} multiplied by w_{21} , H_{03} multiplied by w_{31} plus b . And, if you substitute the numerical values so, we will be getting the numerical value something like this and this is nothing, but the input of your the output layer.

(Refer Slide Time: 21:39)

The output of the network is found to be as follows:

$$O_{01} = \frac{e^{a_2(O_{11}+b)} - e^{-a_2(O_{11}+b)}}{e^{a_2(O_{11}+b)} + e^{-a_2(O_{11}+b)}} = \underline{0.981265}$$

Now, the target output for this training case, is equal to 0.9 (refer to the table). Thus, the deviation in prediction is found to be equal to $0.9 - 0.981265 = \underline{-0.081265}$

And, on the output layer we have got some transfer function so, using that so, very easily we can find out what is your O_{01} that is nothing, but the output of the first neuron lying on the output layer and we can use this your the tan sigmoid transfer function and it will be getting the calculated output is nothing, but 0.981265.

Now, this calculated output will have to compare with the target output and this target output is nothing, but is your 0.9 and we can find out the deviation in prediction is nothing, but 0.9 minus 0.981265, so, this is actually nothing, but the deviation. So, the deviation is coming to be negative here. So, deviation is minus 0.081265. Now, this is what happens after passing actually the first training scenario. Similarly, we are going to pass the second training scenarios, third training scenarios and all capital L training scenarios we are going to pass one after another.

Now, this particular deviation it could be either positive or negative and that is why actually what we do is we try to consider the mod value of these particular your deviation and the mod value of this deviation will be positive. So, for each of these particular the training scenarios or the training cases, we will be getting some deviation and then we find out what should be the average deviation and that particular average deviation will be the fitness of the GA and as we discussed that for each of the GA string. If we can find out the fitness information now we can use the operators like reproduction crossover and mutation. And, GA through a large number of iterations we will try to

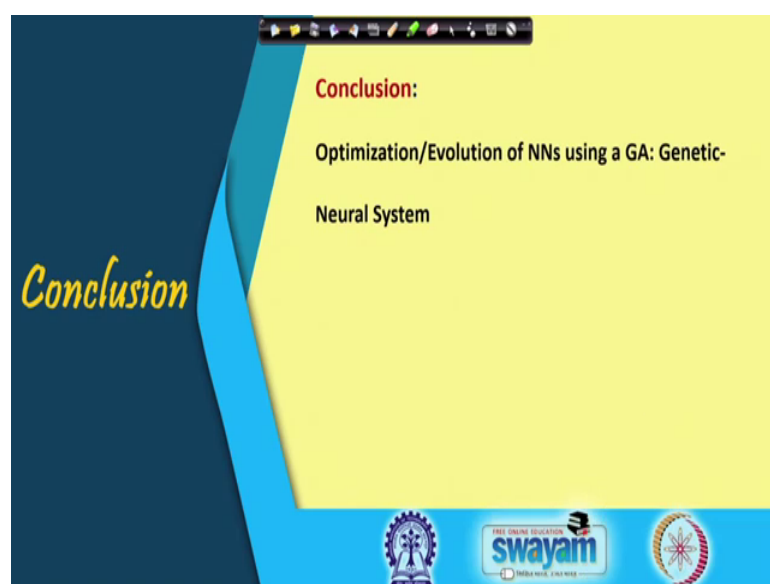
actually evolve so, that particular network so which can predict the input output relationship in a very accurate way.

(Refer Slide Time: 24:08)



Now, if you see the reference like whatever we have discussed on genetic neural system the same thing has been discussed in detail in the textbook of this particular course Soft Computing: Fundamentals and Applications, written by me. So, this is the reference for this genetic neural system.

(Refer Slide Time: 24:34)



And, now to conclude or to summarize whatever we have discussed, we have discussed the principle of the genetic neural system whose main purpose is to evolve a suitable neural network which can predict the input-output relationship of a process very accurately. Now, the working principle we discussed in details and after that we have solved one numerical example to make it more clear and I hope that you have understood the working principle of genetic neural system and by solving the numerical example so, this particular concept has become more clear.

Thank you.