Traditional and Non-Traditional Optimization Tools

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

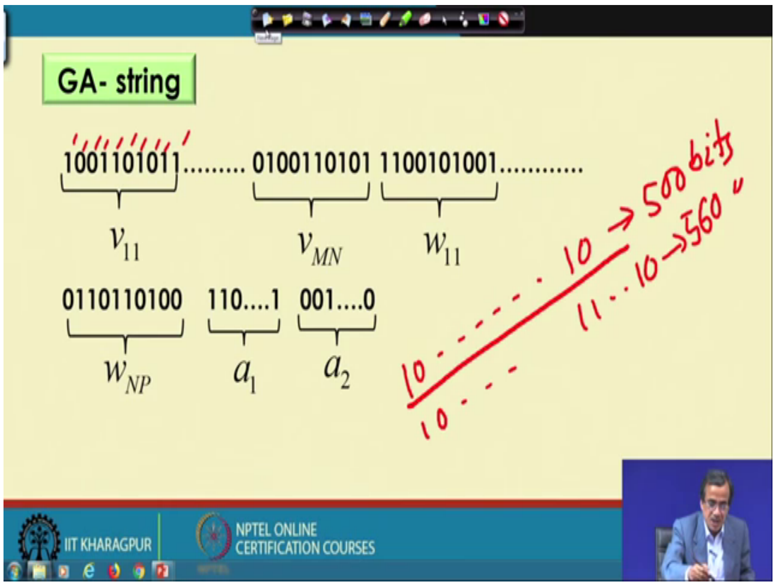
Lecture - 39
Genetic Algorithm as Evolution Tool (Contd.)

Say, we have decided to use a genetic algorithm to evolve a suitable neural network which can make the prediction of the input output relationship very accurately. And we have already seen that the performance of a network depends on the connecting weights, the coefficient of transfer function and the bias value if you use. Now, if you want to optimize, so this particular network we will have to find out the optimal values of these particular the design variables using the genetic algorithm.

There is another way of optimizing, so this particular network that is called the topology optimization of this particular the network. Now, this topology network optimization problem I am not going to discuss for the time being because that will take more time. So, what I am going to do is, so I am just going to solve like how to determine the optimal values of this particular the parameters the design parameters like your the connecting weights, the coefficient of transfer function and bias value if any. So, how do how to code all such design variables in the GA string.

Now, I have got both the options like I can use binary coded GA I can also use real coded GA because all the design variables are real in nature. Now, here actually what I am going to do, I am going to show you how to use a binary coded GA to represent so this particular the problem; that means, the whole network, so how to represent using a particular the GA string.

(Refer Slide Time: 02:05)



GA- string

$\underbrace{1001101011}_{v_{11}}$ .... $\underbrace{0100110101}_{v_{MN}}$ $\underbrace{1100101001}_{w_{11}}$ ....

$\underbrace{0110110100}_{w_{NP}}$ $\underbrace{110....1}_{a_1}$ $\underbrace{001....0}_{a_2}$

10 . . . . . . 10 → 500 bits
11 . . 10 → 560 "
10 . . .
NPTEL ONLINE CERTIFICATION COURSES
IIT KHARAGPUR

But before that as I told that I will have to assume that it has got a fixed topology; that means, your the number of the hidden layers and number of neurons lying in the hidden layer those are known those are predetermined. Like if I keep those things as variable we are going to face more difficult situation that situation I am just going to tell you after some time, but before that let me try to concentrate on this. That means, I am not going for topology optimization, I am just going to find out the optimal values of this particular the desired variable. But truly speaking we will have to find out the optimal topology at the same time the optimal values of this particular the design variables. So, both the things are to be done, but for simplicity I am not going for the topology optimization for the timing.

Now, here as I told that I am going to use a binary coded GA to code all the design variable and the principle of binary coded GA we have already discussed in much more detail. So, I am not going for that. The only thing which I will have to decide the number of bits to be assigned to represent a particular variable and that depends on the required precision in the values of the variable. For example, say here I am just going to consider like 1 2 3 4 5 6 7 8 9 10, like 10 bits to represent each of these particular the connecting weights like all the V values, each of all the V values and each of all the W values to represent I am using 10 bits.

And similarly to represent the coefficient of transfer function a 1 and a 2. So, I am using a few bits here I am using a few bits here. Now, if I just add them up, so this particular GA string will be very long and as you know that if in the binary coded GA if I have got a very lengthy other string, so it is going to give some sort of problem and that problem I have already discussed and that is known as the permutation problem.

So, I am not going to repeat that only thing I just want to mention that if you have got a large number of design variables here and for each variable if I use 10 bits 10 bits. So, there is a possibility. So, this particular the GA string will be very long and this particular representation may may not be very efficient.

Now, what is the alternative? The alternative is we can go for the real coded GA or all the variables will be represented using the real values. And I am not going for that I am just going to concentrate on this binary coded GA at that to on a very simple problem; that means, I am not going to consider the topology the optimization of this particular the network.

Now, that particular problem this topology optimization problem which I am not going to discuss here is bit complicated and just to understand the complexity of that particular problem let me tell you let me explain little bit. Now, supposing that I am going to design and develop a network not only its design variables, but also its topology also we are going to optimize. So, for these design variables we will have to assign some bits and the topology which is decided by the number of layers number of hidden layers, and the number of neurons in the hidden layer that is going to decide actually what should be the total number of this particular the connecting weights and might be if you use like more than one hidden layer. So, there could be here a few more this thing like your, the coefficient of transfer function.

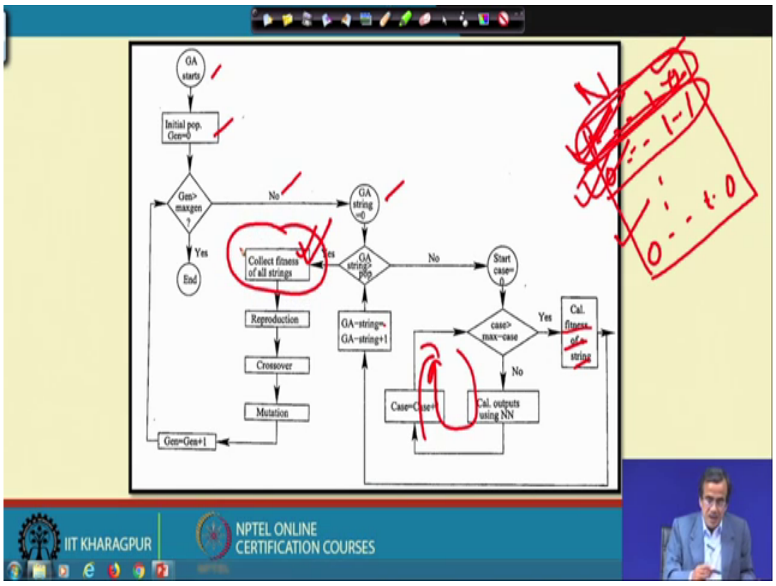
So, the number of variables are going to increase like anything, this is one problem. But there is another very serious problem like corresponding to a particular topology of the network and if I want to represent the topology as well as the design variable might be let me take a very hypothetical example might be I need one binary codes. So, the length could be say 500, there are 500 bits for a particular solution. Might be these 500 bits is going to represent both topology as well as the design variables.

Now, I am just going for the second GA string. Now, once again if I keep the topology as one of the decision variable then the number of hidden layers is also going to vary; that means your the number of the neurons are going to vary; that means, the connecting weight the number of connecting weight is going to vary. Might be in the second GA string the number of bits could be like your is more than 500 might be there could be 560 bits.

So, the number of bits in the different GA string in a particular population of the GA will not remain the same and that will create problem during this particular your the crossover operator. So, I am not going to discuss that particular problem in much more details. To solve that type of problem in fact, we will have to go for a special type of genetic algorithm that is called the messy genetic algorithm. So, I am not going to discuss that particular thing in detail.

So, what I am going to do is I am just going to take a very simple problem having the fixed topology of the network; that means, the number of bits which I am going to assign to represent the different the bits in the different GA string, so the string length will remain the same. And I am just going to find out with the help of GA what should be the optimal network and how to evolve that particular the optimal network. So, that problem the simplified version of that particular problem I should say I am going to discuss. So, this is the way actually we will have to represent the different variables in the GA string.

(Refer Slide Time: 09:12)



And once you have got this particular representation, so we will have to follow this particular the flowchart to implement, this particular algorithm, so that the GA can evolve that particular the network. Now, let us try to understand, so this particular the flowchart.

Now, here as I told the information related to the multi layered feed forward network that will enter inside the GA that is inside the objective function of the GA. And supposing that I am just going to start with start this particular GA, so GA starts. So, we assign that number of generation that is equals to 0 or the iteration equals to 0 and here we have got one check where this number of generation whether it is going to exceed the maximum number of generation pre specified by the user. So, if this particular condition fulfills so that will be the end of this particular program otherwise if it is no. So, we come here and in the population actually we have got a large number of the GA string.

For example, saying if I consider. So, this is the population of GA. So, we have got actually a large number of GA string here. So, we have got a large number of GA string here and corresponding to a particular GA string. Now, corresponding to a particular GA string, so my neural network is ready it is it has got a fixed topology and its connecting weights the coefficient of transfer function; that means, the whole information of the network is ready with me the neural network is ready with me.

So, corresponding to a particular GA string, so what we do is we send all the training scenarios. So, we have got a large number of training scenarios that is nothing, but the known input output relationship might be 500, 1000 something like this. So, depending on actually the number of design variables will have to decide the number of training scenarios. Now, the number of training scenarios should not be less than the number of unknowns. So, the number of training scenarios should be either equal to the number of unknowns or slightly more.

So, this particular the training scenarios, so I am just going to pass one after another through this particular network which is indicated by the first GA string. So, correspondingly the first GA string my neural network is ready and I am passing all the training scenarios one after another.

So, start case equals to 0, the first case. Now, here there is one check whether the case is greater than equals to the maximum case maximum case means the total number of training scenarios which I am going to pass. So, if it is no, then what happens is, so this process will go on and go on so case is case plus 1. So, this is the counter. So, the number of case, number of training scenario is going to increase and I am just going to pass all the training scenarios.

So, let me repeat corresponding to the first GA string my network is ready and through this network I am passing all the training scenarios, so all 500 training scenarios one after another. And once I have passed all the training scenarios; now I can find out, so what is the output for each of these training scenario. I can sum them up and I can find out what should be the fitness of this particular the GA sting. Now, how to determine this particular fitness I will be discussing in the next slide.

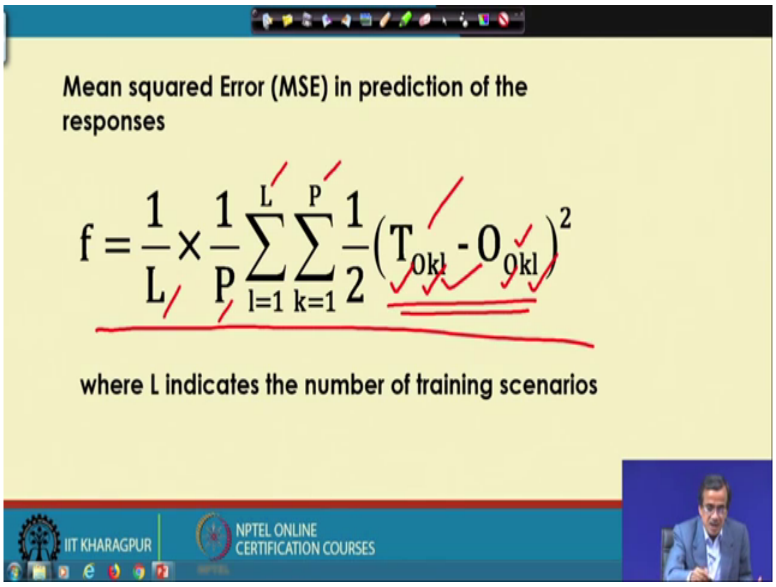
But for the timing let me tell you that this particular is actually whatever output we are getting, so using the output and by comparing the target value we will try to find out the error in prediction and this shows actually the mean squared error in prediction. So, the fitness is nothing, but the mean squared error which I am going to discuss in the next slide. So, for the time being let me assume that the fitness of the GA string is nothing, but the mean squared error in prediction.

Now, once you have got the fitness of a particular GA string. So, for this particular GA string the fitness is known which is nothing but the mean squared error in prediction. Now, actually I go for the second GA string. So, here, this particular, this particular information of the fitness, it will come here and here it is GA string equals 2 GA string plus 1. So, I am just going to send the second GA string and once this second GA string is ready once again I am going to pass the all the training scenarios. So, for each training scenario once again I will be getting the output ok, and you find out how much is the error you sum them up, and you will be getting this particular fitness which is nothing, but the mean squared error.

So, for the second GA string I will be able to get. So, that particular fitness value and all such fitness values you just collect here. So, all the GA string are will be passed to one after another and corresponding to a particular GA string all the training scenarios will be passed and by following that so we will be getting the whole population and its fitness information for this particular the GA. And remember one thing once again if this is the population of the GA string, so each population is going to represent one network and its corresponding the fitness value I have collected here.

Now, if you see this particular, so it looks like this.

(Refer Slide Time: 15:49)



Mean squared Error (MSE) in prediction of the responses

$$f = \frac{1}{L} \times \frac{1}{P} \sum_{l=1}^{L} \sum_{k=1}^{P} \frac{1}{2} (T_{Okl} - O_{Okl})^2$$

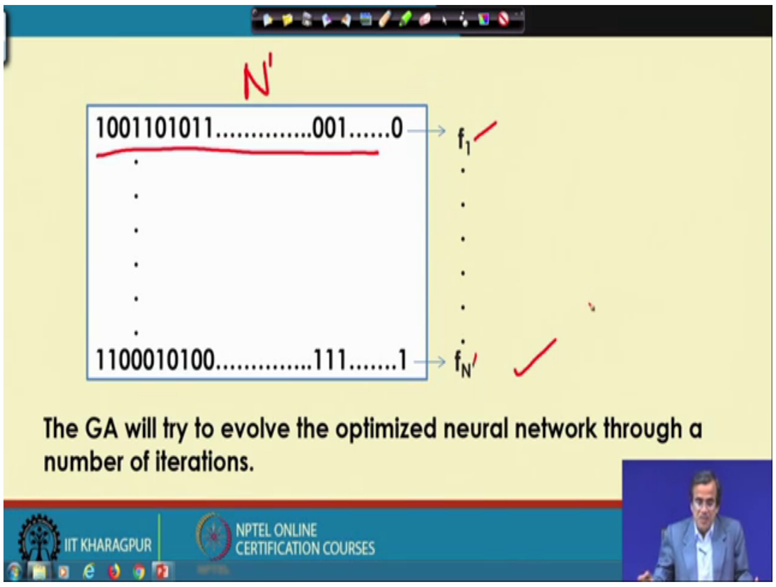
where L indicates the number of training scenarios

So, this is the way actually we can find out the mean squared error for each of the GA string. So, we will have to calculate the mean squared error that is one divided by L multiplied by one divided by P summation L equals to 1 to L summation k equals to 1 to P half, T Okl minus O Okl square.

Now, what is this T Okl? T Okl is corresponding to lth training scenario and corresponding to the kth output neuron. What is the target output? And this O Okl is nothing, but the output of the calculated output of the kth neuron lying on the output layer corresponding to the a lth training scenario and this particular difference is going to give that particular the error. The square of that just to make it positive and this summation k equals to 1 to P, P indicates the total number of the outputs on the output layer, capital L indicates the total number of training scenarios like the training cases and this P is nothing, but is actually the number of output on the output layers number of output neurons on the output layer and capital L is nothing but your, the training scenarios.

So, using this particular expression, I can find out the mean squared error in prediction at this particular the mean squared error in prediction is nothing, but actually what I mentioned as your, this particular the fitness. So, I will be getting this particular the fitness information. And once you have got this particular fitness information here, so now, we are in a position something like this.

(Refer Slide Time: 17:47)



N'

1001101011..............001......0 $\rightarrow f_1$

.

.

.

1100010100..............111.......1 $\rightarrow f_{N'}$

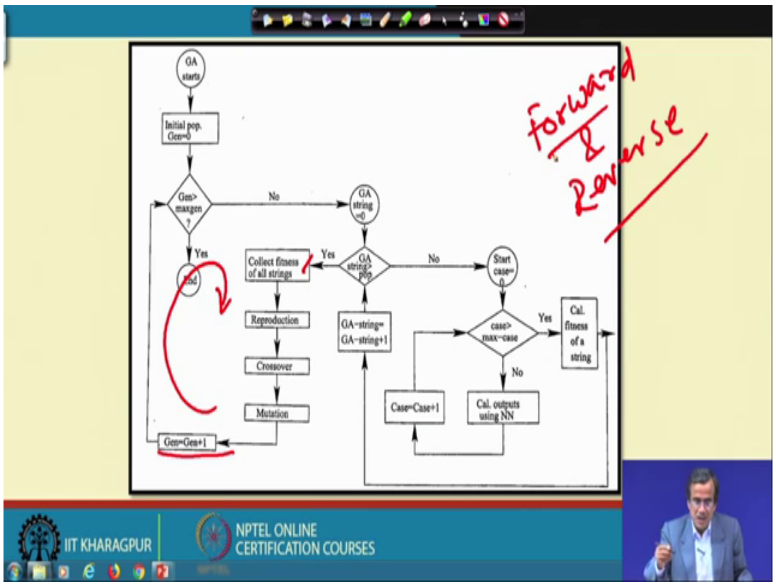
The GA will try to evolve the optimized neural network through a number of iterations.

Say this is the population of the GA, the binary coded GA the total size is N prime and here, so corresponding to the first GA string supposing that I have got the fitness information that is nothing, but the mean squared error in prediction.

Similarly, corresponding to the second third up to the last one that is the N prime. So, corresponding to this I can find out that this is the fitness. And once you have got this particular fitness information. Now, we can go back we can go back to this particular the flow chart. So, we are here, so we are here, all the fitness information are known for this whole population. Now, we go for the reproduction scheme, just to get the mating pool, we go for cross over for exchange of properties and we go for mutation part sudden change of this particular the parameters and this is the generation counter, generation equals to generation plus 1 and this process will go on and go on and through a large number of iteration or the generation the GA is going to evolve so that particular the network.

So, the using this particular evolve network now if I pass the set of training scenarios the test scenarios the test cases which are new which are not used during the training there is a possibility that it will be able to predict the output accurately and as I told that this particular principle can be used both for the forward mapping and your both forward and this reverse mapping. And this problems are to be solved the forward to the problems are both forward as well as the reverse mapping are to be solved because we want that input output relationship in both the direction like if you want to automate any such engine process or any such engine system.

(Refer Slide Time: 19:40)



Now, this method has got a few drawbacks I should say, one drawback I have already mentioned. For example, say if I use a the binary coded GA and if it involves a large number of variable. So, it is going to suffer from that type of your the permutation problem which I have already discussed in details. So, that is why as I told we switch over to the real coded GA and with the help of this particular real coded GA there is a possibility that we will be able to evolve that particular the network.

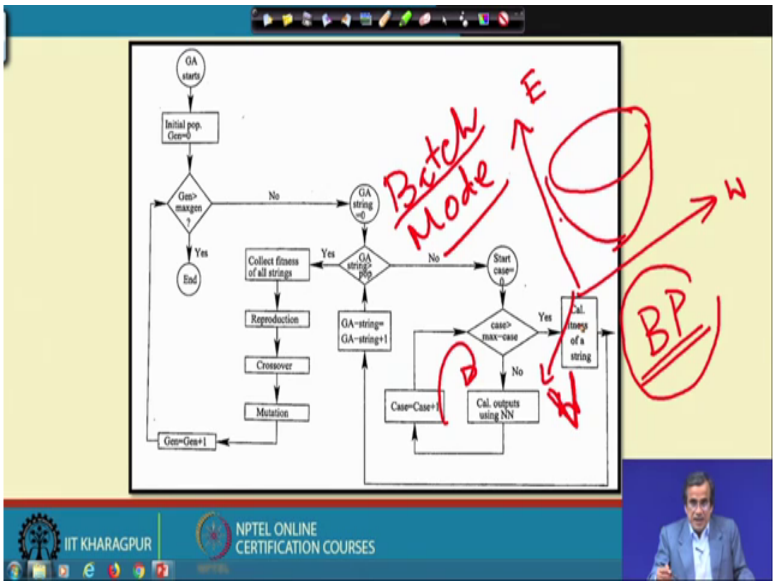
Now, if I compare, this particular network with the network which I could get using the principle of the traditional optimization like almost simulators the steepest descent method. Now, and if I compare the network evolved using that steepest descent method and the network evolved using this particular the genetic algorithm, now which one is better.

Now, our experience says actually as the steepest descent method works based on the principle of the gradient there is a possibility of local minima. Now, if I use this genetic algorithm to evolve this particular network there is a possibility that we may not face that type of problem and it will be able to evolve a very efficient network which will be able to make this particular prediction very accurately. But our experience says that there is no guarantee that every time, so this particular the GA tuned or GA evolved network is going to give better results compared to the back propagation algorithm or the steepest descent algorithm.

Now, the reason is very simple. The reason is actually it depends on the nature of this particular the error function. Now, for most of the problem like whenever we are going to train a particular network with the help of either say steepest descent algorithm or the back propagation algorithm which I have not discussed, and this back propagation algorithm I have not discussed, but if I use genetic algorithm there is absolutely no guarantee that every time you will be getting better results compared to the your back propagation algorithm. And the performance of the network depends on the nature of the error function.

Now, if you see the error function which I have already considered which I already do it here. Now, if you see this particular error function for example, say if I consider, so this is this is say W, and this is V, this is say V, and this is the error now, there is no guarantee that every time you will be getting a very well defined error surface and very unimodal error surface.

(Refer Slide Time: 23:12)



Now, if you find a very unimodal function for example, say this could be one unimodal function sort of thing ok, this is a very regular. Now, fortunately if I get this type of error function during the training of the network. So, the back propagation algorithm which I have not discussed which works based on the steepest descent algorithm that could be even better compared to the GA tuned or the GA evolved network.

On the other hand supposing that we have got a very complicated error function and the error function is having multimodal and there are some discontinuity in this particular your the error surface. Now, in that case there is a possibility that this GA evolved network will perform better compared to this type of so BP tuned the neural network and that is why actually the concept of this network evolution of neural network using the principle of genetic algorithm could reach much popularity. And as I told I have already told that in robotics particularly in the field of intelligent robotics one new field has emerged which is the evolutionary robotics where our aim is to evolve.

Once again let me repeat, soour aim is to evolve the suitable adaptive motion planner adaptive controller so that this particular the intelligent robot can perform in a very efficient way and that is why there is a trend for the people who are working in the field of intelligent robotics. So, there is a trend to switch over to this particular area like how to how to tackle the situation in an unknown scenario and how to take the situation as the situation demands. And if we just go for this type of GA evolved neural network there is a possibility this particular neural network will have some capability and that is called some sort of adaptive capability.

Now, by this adaptive capability we mean that during the training scenarios, during the training period if you do not use exactly those training scenario and if the test scenario is significantly different from the training scenario then also there is a possibility that this particular network will be able to provide some effective results, some suitable results. The reason is actually here during the training of this particular the network we have used a particular training algorithm that is called the batch mode of training.

Now, in this batch mode of training the batch mode of training we use just providing the information of all the training scenario at the moment we consider the average error or we try to find out your that that mean square error, root mean that mean square error and the moment we consider the average effect of this particular error. So, there is a possibility that we are going to inject some sort of adaptability to this particular network and that is why the GA tuned or the GA evolved network will be very adaptive.

Thank you.