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

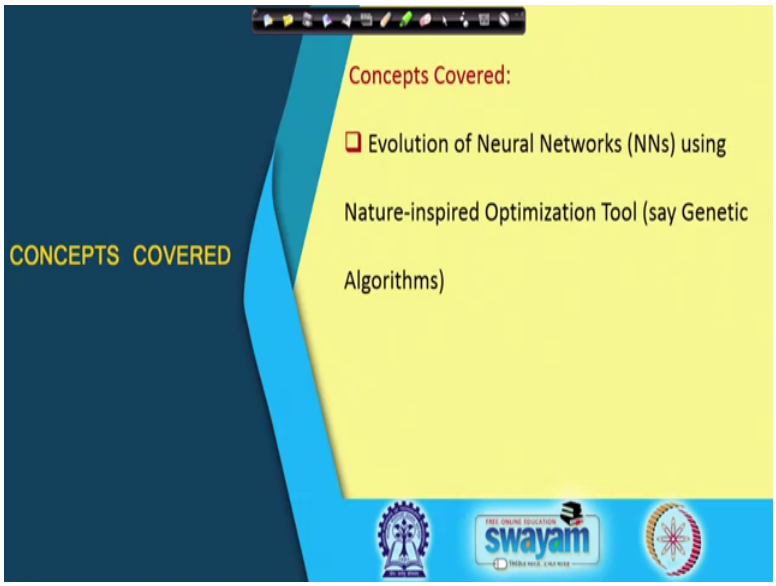
Lecture - 31
Optimal Designs of Neural Networks

Now, we are going to discuss on the topic Optimal Design of Neural Networks. Now, we have seen that we can use a multi layered feed forward network to model input output relationship, of an engine system, of an engine process, accurately both in the forward as well as in the backward directions. But, this particular network does not have an inbuilt optimization tool. Now, what do you will have to do is to ensure the accuracy in prediction. So, we will have to use an optimizer along with this particular network in order to train it.

Now, what he can do we can use some traditional tools for optimization, just to optimize this particular network, like we can use steepest decent our algorithm. In the form of the back propagation algorithm and we have already discussed so, this type of network is very popularly known as the back propagation neural network. Now, on the other hand we can also use some set of nature inspired optimization tools, like genetic algorithms, particle swarm optimizations and others, just to optimize or to evolve the neural networks, which will be able to predict the input output relationship, very accurately both in the forward as well as reverse direction.

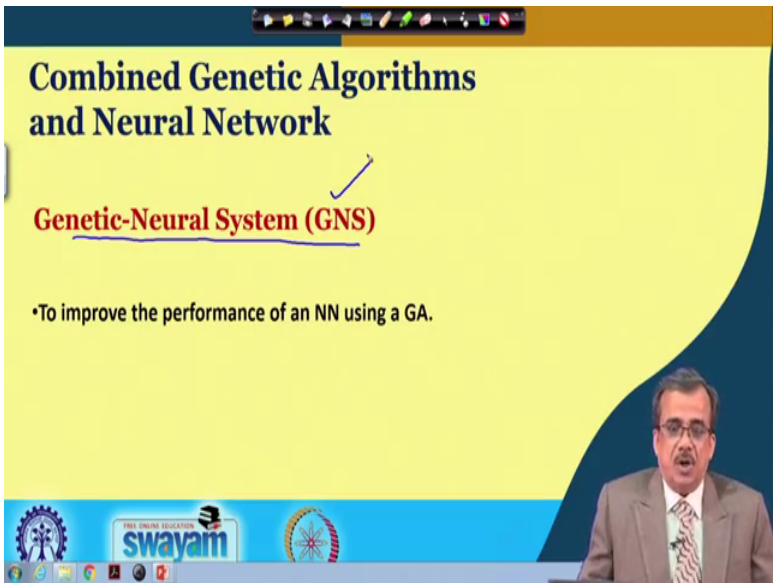
Now, here in this topic I am just going to discuss like how to evolve a multi layered feed forward network, using the principal the genetic algorithm based principle of evolution.

(Refer Slide Time: 02:25)



So, if you see the topic which I am going to discuss let me repeat once again. We are going to discuss the principle of evolution of neural networks using a nature inspired optimization tool say genetic algorithm. Or, in other words we are going to optimize the performance of this particular neural network using one nature inspired optimization tool say genetic algorithm.

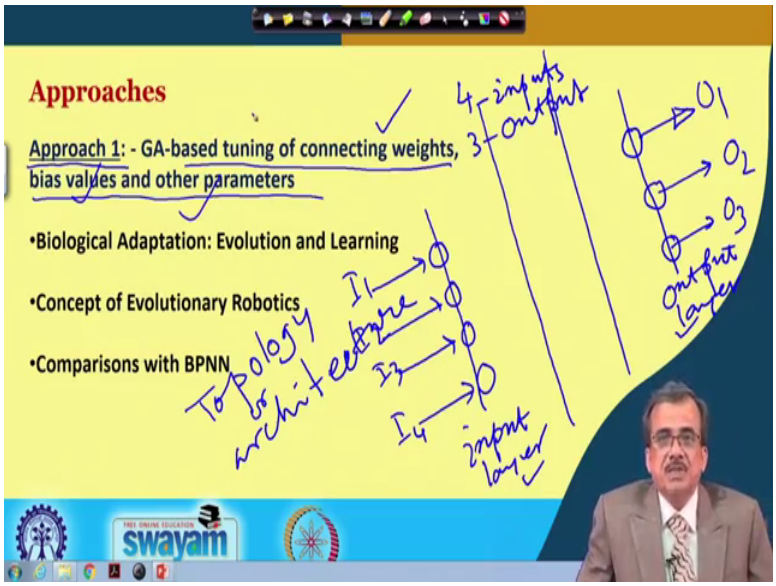
(Refer Slide Time: 02:56)



Now, if you see the combined genetic algorithm and neural networks. Now, this is very popularly known as the Genetic Neural System and in short this is nothing, but the GNS.

Now, the purpose of developing this genetic neural system is to optimize the performance of a neural network with the help of a genetic algorithm. Now, let us see how to optimize the performance of a network using the principle of evolution of this genetic algorithm.

(Refer Slide Time: 03:44)



Now, if you see the performance of a network it depends on actually a number of things. For example, it depends on the topology of this particular network, it depends on the connecting weights; that means, the connecting weight between the input and hidden layer. And, that between the hidden and output layers, it depends on the bias values, it depends on the coefficient of transfer function of the different layers.

Now, supposing that I am just going to use a multi layered feed forward network to model input output relationship of a process. And, let me take a very simple example supposing that the process is having say 4 inputs. So, it is having 4 inputs and there are say 3 outputs of this particular process and I want to model, its input output relationship with the help of one multi layered feed forward network.

Now, if I concentrate on the input layer. So, there will be 4 neurons. So, these are the neurons on the input layer, say I have got 4 inputs like I 1 I 2 I 3 and I 4 and there are 4 outputs so on the output layer. So, there will be actually your the 3 out 3 outputs so, that is nothing, but O 1, O 2, and O 3.

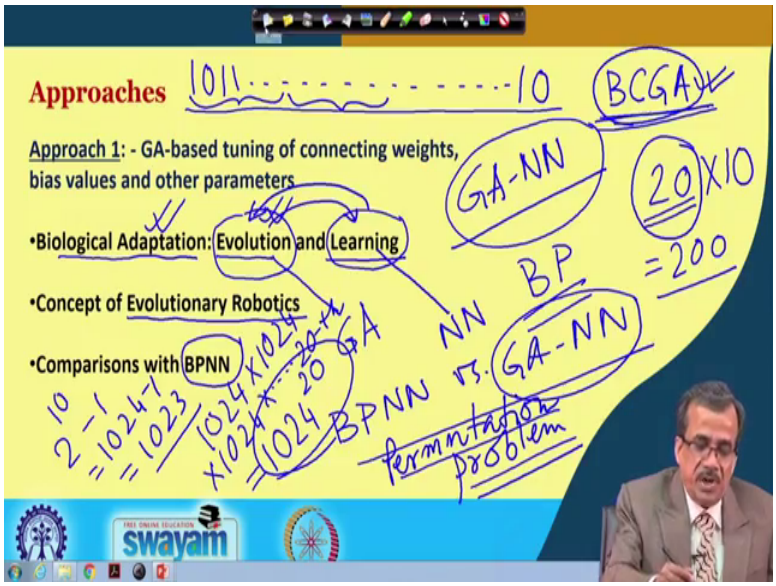
Now, this is the input layer. So, this is nothing, but the input layer of this network at this is nothing, but the output layer of this particular the network. Now, the number of neurons to be present on the input layer and that to be present on the output layer are kept fixed to the number of inputs and the number of outputs of the process to be model respectively; that means, your on the input layer. So, there will be 4 neurons; on the output layer there will be actually 3 neurons.

Now, the topology or the architecture of this particular network depends on how many hidden layers we put in between the input layer, and the output layer and how many neurons which we are going to put on these particular your the hidden layers. Now, actually that is going to decide what should be the topology or architecture of this particular the network. Now, the performance of this particular network largely depends on the topology of these particular the network. Now, here actually we are going to discuss two different approaches to develop the genetic neural system.

Now, I am just going to concentrate for the time being on approach 1 and that is nothing, but genetic algorithm based tuning of connecting weights, bias values and other parameters. That means, in approach 1 we are going to consider a network having known or the fixed topology or the architecture. So, this is actually nothing, but a network of fixed topology or architecture. Now, if you just keep the architecture or the topology of the network fixed. Now, its performance depends on the connecting weights, it depends on the bias values, it depends on the coefficient of transfer function and so on.

So, our aim in approach 1 is to determine the optimal values of this particular network show that it can perform in the optimal sense. Now, before I go for the principle of that particular the approach 1, I just want to discuss one fact from the biological adaptation. Now, if you see the principle of biological adaptation.

(Refer Slide Time: 08:22)



So, in biological adaptation actually there are two things; one is your the principle of evolution and we have got the principle of learning. Now, this evolution and learning are two important parameters in biological adaptation. Now, the same thing actually we are going to copy it here in the artificial way. Now, before I copy it so, let us discuss what is happening in biological adaptation.

Now, if you concentrate on, so, this particular learning. Now, learning takes place during once lifetime on the other hand the evolution takes place through a large number of generation or a large number of iterations. So, this evolution and learning so, these two parameters are working on two different scales timescales.

Now, if we concentrate on this particular the learning. Now, we go on learning. So, long as we live in this particular world and we try to collect good information, good knowledge, now this particular the good information and good knowledge we want to pass it to the next generation. Now, if we pass it to the next generation, there is a possibility that is going to accelerate the rate of evolution, because the next generations are going to get all the good information. So, there is a chance the rate of evolution is going to increase.

Now, on the other hand if you see, the principle of learning. So, during the learning actually one spends a lot of time on carrying out the optimization; that means, for this learning knowingly or unknowingly we use the principle of optimization. And, if you see

the optimization tool particularly the nature inspired optimization tool. So, in these tools actually we use the principle of evolution. So, this particular evolution is going to help learning and learning is also going to help your the evolution. So, they are helping each other and through this particular mutual help, there is a chance. The rate of biological adaptation is going to increase the same thing has been copied here so, in this particular the genetic neural system.

Now, what we do is we use some evaluation tool like say genetic algorithm say denoted by GA-NN we use some learning tool for example, say it could be a neural network and in this combined tool so, this particular GA and neural network. So, what we do is we try to optimize the neural network, or we try to improve the performance of the neural network, and we try to actually design and develop. What is known as the combined, GA neural network technique and that is nothing, but genetic neural system.

Now, there are many applications of this particular the genetic neural system. For example, say in the field of robotics, there are lot of applications like how to evolve. The adaptive controller for an intelligent robot, how to evolve an adaptive controller for a particular the motor used in robots. Now, so, the principle of evolution has been used in robotics and this works based on actually the principle of genetic neural system, and a new field of robotic research has started that is known as the evolutionary robotics.

Now, in evolutionary robotics the main aim is to evolve the suitable motion planner or the adaptive controller instead of going for the direct design of this particular the adapt direct design of the controller or the motion planner. Now, here actually we use the principle of evolution instead of going for the direct design. Now, we actually realize one fact, that through this direct design many things we are unable to foresee beforehand. And, that is why actually we will have to take the principle of evolution for the development of an efficient system.

Now, let me see how to use and how to develop this genetic neural system. Now, I have already mentioned about this GA neural network and I have also discussed about these back propagation neural network. Now, let me repeat in back propagation neural network for optimizing we use the back propagation algorithm that is the BP algorithm. And, this BP algorithm works based on the steepest decent algorithm, this we have already discussed. Now, if I compare the performance of these particular your BPNN and your

the GA NN. Now, this BPNN works based on the steepest decent algorithm; so, there is a chance of the algorithm for getting trapped in the local minima problem. And, the chance of getting the local minima is much less in case of so, this particular the GA neural network.

Now, actually if you see the literature on genetic algorithm, we have got the different versions of genetic algorithm. Now, those things actually I am not going to discuss in details in this particular course, but this is available in the text book used for this particular course. Now, if you see the literature the genetic algorithm the first version of genetic algorithm is nothing, but the binary coded genetic algorithm that is your BCGA. Now, in BCGA actually what you do is all the variables we represent with the help of some binary and binary is nothing, but a combination of 1's and 0's.

Now, here let me take a very simple example supposing that I am just going to optimize a neural network. And, let me assume that there are say 20 variables in this particular the network. So, if I want to optimize the performance of these particular the neural network. So, what do you will have to do is I will have to find out the optimal values for each of these particular the 20 variables. Now, the variables could be the connecting weights bias values the coefficient of transfer function and all such things. Now, here if I is the binary coded GA. Now, to represent each of these particular variables so, I will have to use a number of bits let me assume that, I am going to use say 10 bits to represent each of these particular your the variable.

Now, so, if I just concentrate on a particular GA string there is a binary coded GA string, it looks like this like your say 10 11 dot dot dot and the last term might be say 10. Now, if there are 20 variables and if I use the 10 bits for each variable so, I will have 20 multiplied by 10; that means your 200 bits in one GA string. So, here I have got in fact, 200 bits. Now, out of 200 bits supposing that the first 10 bits are used to represent a particular variable, the next 10 bits are used to represent another variable and so on.

Now, the moment to use 10 bits now; that means, we are going to divide the range of these particular variables into actually 2 raise to the power 10 minus 1, that is your 1024 minus 1, that is 1 0 2 3 equal divisions. Now, for 1 variable so, we have got the 1 0 2 3 equal divisions; that means, I have got say 1024 the numerical values on the range of this particular the variable.

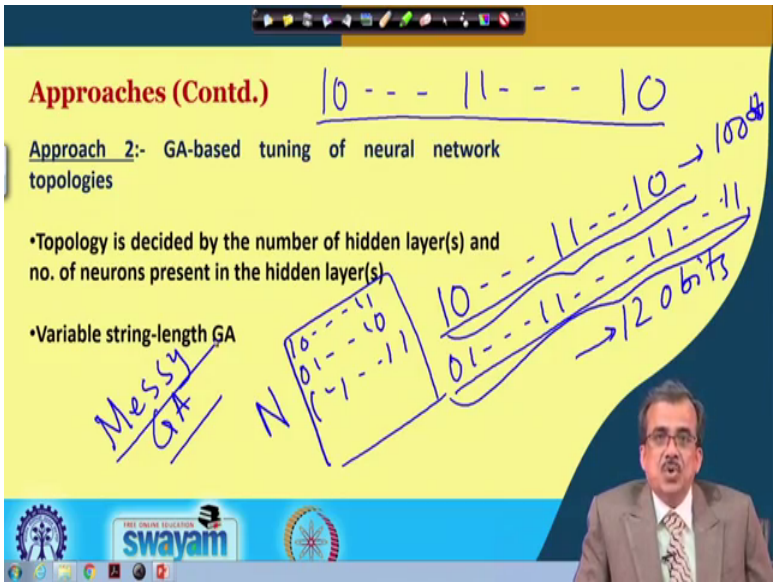
Similarly, on the second variable so, within the range I have got another 1024 numerical values, the third I have got another 1024 numerical values at this will go on up to your say 20th variable. Now; that means, if I see the total number of combinations of the numerical values, which have to be considered before the GA can decide the optimal solution is nothing, but 1024 raise to the power your that 20.

So, this is actually a high number. So, 1024 raise to the power 20 is a high number; that means, the GA will have to carry out some charge. For so, many combinations of the input variables or the design variables, before it can declare, that this is a globally optimal solution. So, this is a very difficult task for the binary coded GA have this problem, in genetic algorithm is known as actually the permutation problems. Now, if there are large number of variables, so that this binary coded GA actually can suffer from, so this particular the permutation problem.

Now, the point which I am just want to make it clear, that if I have got a very large number of variables is better not to use this binary coded GA. And, in place of the binary coded GA impact we can go for some sort of the real coded GA. Now, once again a detailed discussion discussing on a real coded GA is beyond the scope of this particular course, but this thing is available in details in the textbook for this course that is your sub competing fundamentals and applications.

So, we are going to use say the real coded GA just to overcome. So, these particular the permutation problem, but here for simplicity I am just going to discuss this binary decoded GA only; that means, how to use a binary coded GA to design or evolve a suitable neural network. So, that it can predict the input output relationship very accurately.

(Refer Slide Time: 20:24)



Now, let us try to concentrate on approach 1, approach 1 we have already discussed and now I am just going to discuss some numerical examples also after some time. And, let me concentrate little bit on the principle of approach 2. Now, in approach 2 actually what we do is we generally go for the genetic algorithm based tuning of the neural network topology. Now, this I have already mentioned that the topology of this particular network, or the architecture of this particular network, depends on the number of hidden layers and the number of neurons you put under the hidden layer.

Now, if I give this particular task to this genetic algorithm. So, the binary coded GA is going to face one problem the problem is as follows. Now, the number of neurons on the hidden layers and the number of hidden layers so, that has to be encoded inside that particular your the GA string. That means, your so, this particular GA string is going to encode like if I use the binary coded GA. So, this is nothing, but a particular GA string. So, it is also going to encode the information related to the number of hidden layers, which we are going to use and the number of neurons to be present in this particular the layer.

And; that means, your so, from 1 GA string to the next GA string the number of hidden layers and the number of neurons to be present in the hidden layer are going to vary; that means, we are going to face a problem that is called the variable string length genetic algorithm. Now, let me explain supposing that in the population of GA, we have got a

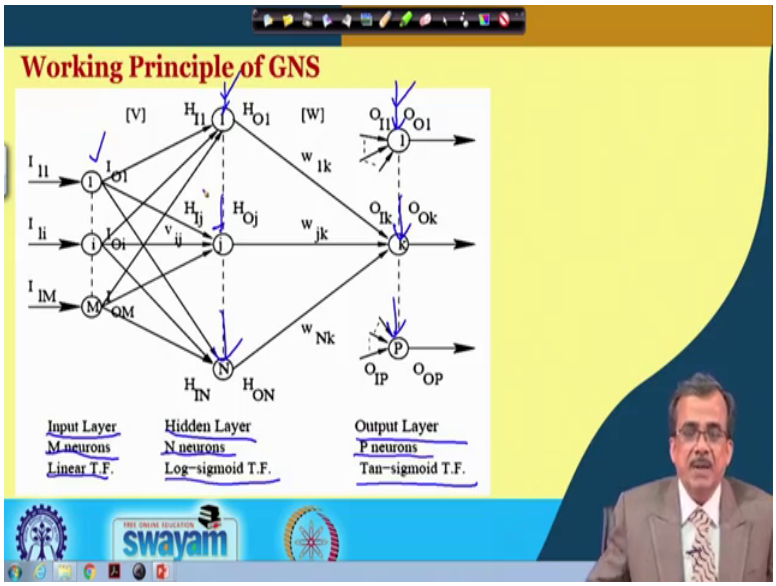
large number of solution and whose side is denoted by N, that is the population size, that will see here I have got all such binary string say. So, these are all binary strings here. Now, here if I consider that your this approach to whose aim is to optimize the topology or architecture of the network.

So, there is a possibility that I will be getting one GA string might be it is having the length say 100. So, the GA string may look like this. So, this is there are say 100 bits here say 100 bits at the next GA string it may have say 120 bits. So, there could be another GA string here, and which may have say 120 bits. Now, 1 GA string is having 100 bits another is having 120 bits.

Now, if I just go part the crossover operator, now for this particular binary coded GA. So, we are going to face a lot of problem, because so, this particular GA string is having 100 bits and this particular GA string is having 120 bits. So, for the last 20 bits actually we cannot easily do the crossover operation. So, we are going to face a lot of problem in crossover particularly, if you use the binary coded GA.

Now, that is actually a problem related to the variable sting then genetic algorithm. Now, there are some ways to overcome this particular problem and we have got a special type of GA, that is called your that messy GA. The messy GA is another very popular GA, where we consider the variable string length during the crossover operation. Now, this is once again beyond the scope of this particular course. So, this messy GA actually I am not going to discuss in details.

(Refer Slide Time: 24:40)

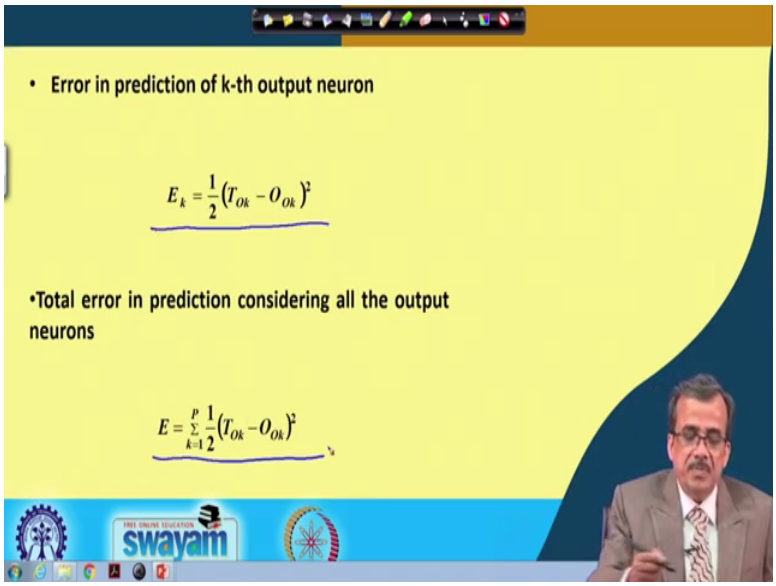


Now, let us try to concentrate on these multi layered feed forward network. Now, as I discussed several times like your. So, this network is having your say 3 layers, like your the input layer, then hidden layer and the output layer. Now, this input layer is having M neutrons, the hidden layer is having N neurons, and the output layer is having P neurons. And, let me use the linear transfer function on the input layer the log sigmoid transfer function on the hidden layer and tan sigmoid transfer function on the output layer.

Now, this particular network I want to optimize or this particular optimal network we want to evolve with the help of a genetic algorithm. So, what you will have to do is. So, all the design parameters, like the connecting weights between the input layer, and the hidden layer, and the connecting weights between the hidden layer, and the output layer we will have to optimize. We will have to optimize the coefficient of the log sigmoid transfer function, the coefficient of tan sigmoid transfer function, and for simplicity I did not consider any bias value here and if I consider the bias value for example, if I consider the bias value. So, those buyers various also we will have to optimize.

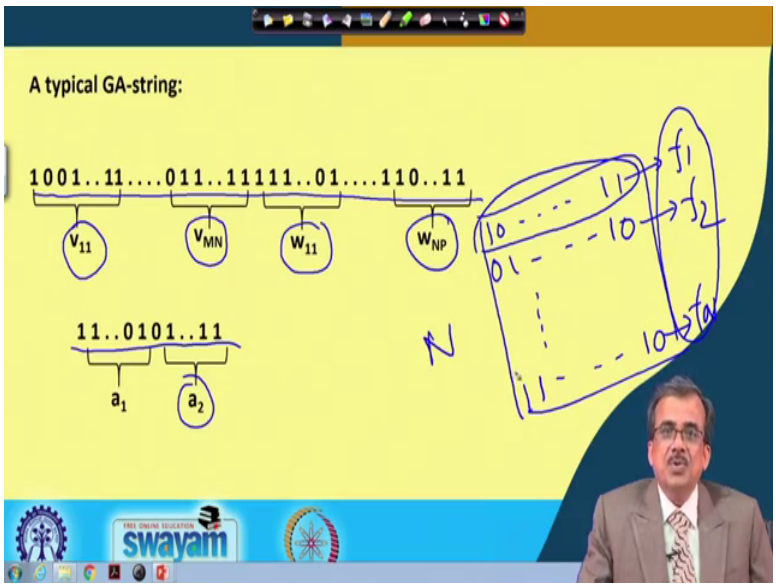
And, the working principle of this multi layered feed forward network. I have already discussed in details. So, I am not going for that once again. Now, instead what I am going to discuss how to represent this particular network inside 1 GA string.

(Refer Slide Time: 26:37)



Now, before I discuss that let me recapitulate that error in prediction of k-th output neuron is nothing, but is your so, this particular expression that is E k is nothing, but half T minus O square. Then, the total error in prediction considering all the output neurons is nothing, but E that is summation k equals to 1 to P half T minus O square. So, by using this we can find out what is the total error in prediction.

(Refer Slide Time: 27:25)



Now, here actually our aim is to represent. So, this network with the help of a binary coded GA string and we are going to evolve or we are going to optimize so, this

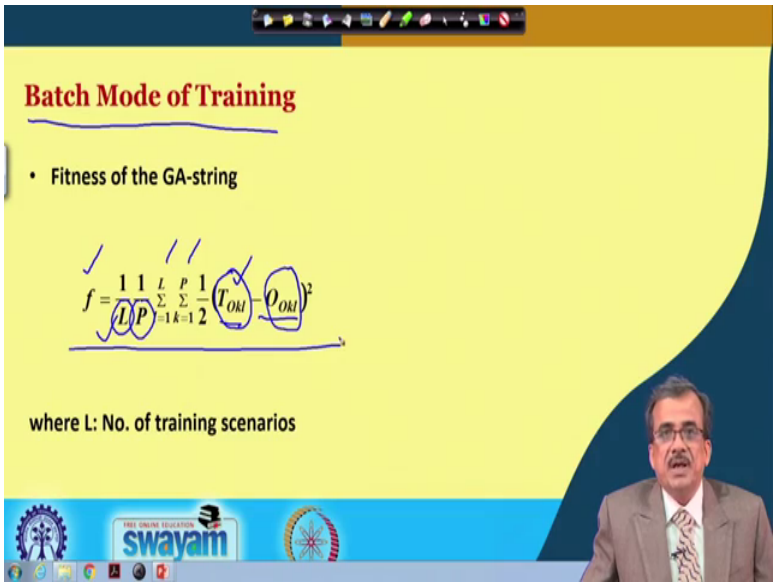
particular the network. Now, this is a particular GA string like the binary coded GA. Now, the first few bits are going to represent what should be the connecting weight, that is V 1 1. And, the last V connecting weight that is V MN is this then the W connecting weights, W connecting weights, now to represent each of these connecting weights. So, I will have to assign a few bits.

Then to represent a one that is the coefficient of transfer function for the log sigmoid transfer function, then a 2 is the coefficient of transfer function for the tan sigmoid transfer function. So, we will have to assign a bits. Now, so, this particular GA string is going to carry information of the whole network and supposing that we know the architecture. And, this optimization we are doing for say the fixed architecture or the fixed topology. Now, if concentrate on say E particular your the GA string. So, it is going to carry the whole information of this particular, the network.

And, similarly in the population of genetic algorithm we have got a large number of strings; that means a large number of neural networks. Now, if you see so, if this is the population of GA string. So, the first GA string could be something like this, second one could be something like this, then the last the nth one could be something like this. So, each of these particular GA string is going to carry information of this particular the network.

And, then as we discussed the principle of genetic algorithm in short, that we will have to find out what should be the fitness, for this particular GA string, what should be the fitness for these particular GA string, the fitness for the nth GA string. And, once you have got this fitness information. Now, you can use the operators like reproduction crossover and mutation and through a large number of iteration the GA will try to find out, that particular network, which will be able to predict actually this input output relationship very accurately.

(Refer Slide Time: 30:15)

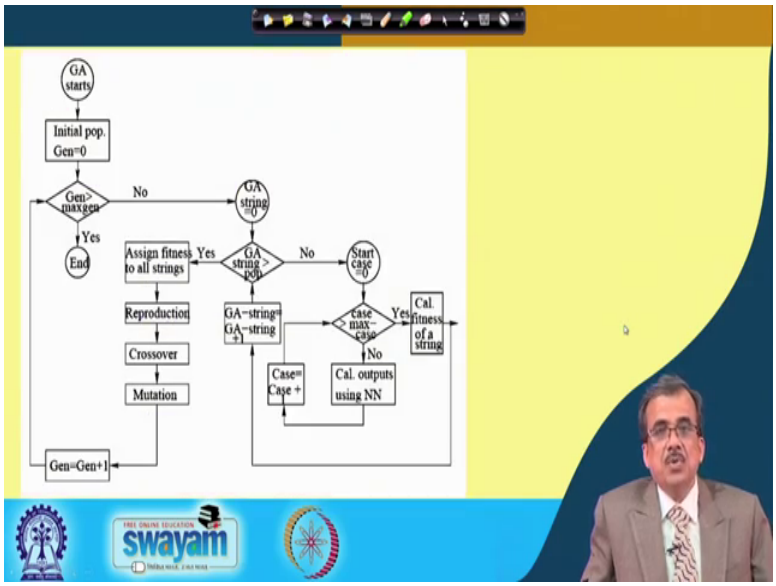


Now, how to define this particular the pit test? Now, to define the fitness actually what we do is we use the concept of actually the batch mode of training. Now, the principle of batch mode of training we have already discussed; that means, we pass a large number of training scenarios denoted by capital L, and we consider all the outputs that is your the capital P number of outputs, and we try to find out so, this average error after passing all capital L training scenarios. And, this is nothing, but the fitness of the GA string is denoted by f is nothing, but one divided by L multiplied by 1 divided by P summation small l equals to 1 to capital L, summation small k equals 2 1 2 capital P half T l minus O l square.

Now, hear this O l is nothing, but the output of the k th neuron lying on the output layer corresponding to the lth training scenario. Similarly, this T l is nothing, but the target output of the kth neuron lying on the output layer, corresponding to the lth training scenario. Now, this is the way actually we calculate the fitness of a particular GA string. And as I told that once you got the fitness information for the whole population; now we are in a position to discuss how it can evolve that optimal network?

Now, here as GA works based on the principle of evolution, now there is a possibility that the GA will try to find out a multiple optimal solutions for this particular the network. Now, if you get the multiple optimal solutions. Now, out of these optimal networks, now anyone actually we can we can use and if use this particular the network.

(Refer Slide Time: 32:43)



So, this network will be able to predict the input output relationship very accurately.

Thank you.