**Computational Method and Technique**

1a) Distinguish between biological neural network and artificial neural network.

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **BNN** | | **ANN** |
| **Processing** | Massively parallel, slow but superior than ANN | | Massively parallel, fast but inferior than BNN |
| **Size** | 1011 neurons and 1015 interconnections | | 102 to 104 nodes (mainly depends on the type of application and network designer) |
| **Learning** | They can tolerate ambiguity | | Very precise, structured and formatted data is required to tolerate ambiguity |
| **Fault tolerance** | Performance degrades with even partial damage | | It is capable of robust performance, hence has the potential to be fault tolerant |
| **Storage capacity** | Stores the information in the synapse | | Stores the information in continuous memory locations |
| **BIOLOGICAL NEURONS** | | **ARTIFICIAL NEURONS** | |
| Major components: Axions, Dendrites, Synapse | | Major Components: Nodes, Inputs, Outputs, Weights, Bias | |
| Information from other neurons, in the form of electrical impulses, enters the dendrites at connection points called synapses. The information flows from the dendrites to the cell where it is processed. The output signal, a train of impulses, is then sent down the axon to the synapse of other neurons. | | The arrangements and connections of the neurons made up the network and have three layers. The first layer is called the input layer and is the only layer exposed to external signals. The input layer transmits signals to the neurons in the next layer, which is called a hidden layer. The hidden layer extracts relevant features or patterns from the received signals. Those features or patterns that are considered important are then directed to the output layer, which is the final layer of the network. | |
| A synapse is able to increase or decrease the strength of the connection. This is where information is stored. | | The artificial signals can be changed by weights in a manner similar to the physical changes that occur in the synapses. | |
| Approx 1011 neurons. | | 102– 104 neurons with current technology | |

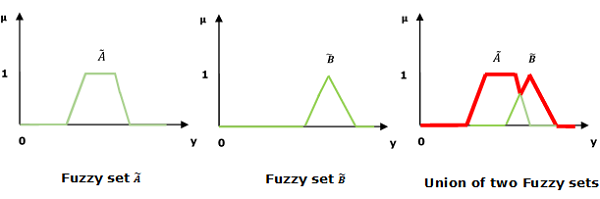
1b) Draw the venndiagram for fuzzy set A and B for

1. Union 2) Intersect 3)Complement

**Union**



Here ∨ represents the ‘max’ operation.

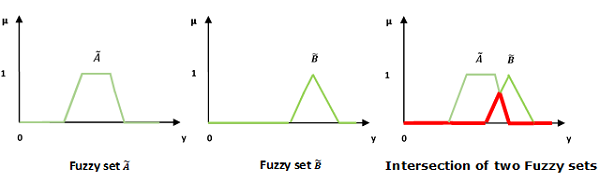


**Intersection**

Let us consider the following representation to understand how the Intersection/Fuzzy ‘AND’ relation works −



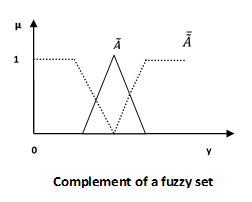
Here ∧ represents the ‘min’ operation.



**Complement**

Let us consider the following representation to understand how the **Complement/Fuzzy ‘NOT’** relation works −





**1c) In BFO elimination and dispersion step, elimination is not considered. Why?**

In the evolutionary process, elimination and dispersal events occur such that bacteria in a region are eliminated or a group is dispersed from current location and may reappear in the other regions due to environmental changes or some natural calamities. They have the effect of possibly destroying chemotactic progresses, but they also have the effect of assisting the chemotaxis, since dispersal may place bacteria near good food sources.

**1d) Briefly explain about ADALINE and MADALINE Network?**

**Adaline** which stands for Adaptive Linear Neuron, is a network having a single linear unit. It was developed by Widrow and Hoff in 1960. Some important points about Adaline are as follows −

* It uses bipolar activation function.
* It uses delta rule for training to minimize the Mean-Squared Error (MSE) between the actual output and the desired/target output.
* The weights and the bias are adjustable.

The key difference between the Adaline rule (also known as the Widrow-Hoff rule) and Rosenblatt's perceptron is that the weights are updated based on a linear activation function rather than a unit step function.

**Madaline** which stands for Multiple Adaptive Linear Neuron, is a network which consists of many Adalines in parallel. It will have a single output unit. Some important points about Madaline are as follows −

* It is just like a multilayer perceptron, where Adaline will act as a hidden unit between the input and the Madaline layer.
* The weights and the bias between the input and Adaline layers, as in we see in the Adaline architecture, are adjustable.
* The Adaline and Madaline layers have fixed weights and bias of 1.
* Training can be done with the help of Delta rule.

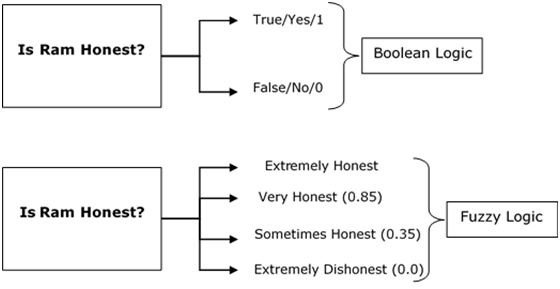
1e) State the factor on which optimization problems concentrate around?

optimization problem is the problem of finding the best solution from all feasible solutions. Optimization problems can be divided into two categories depending on whether the variables are continuous or discrete.

**1f) Justify the following statement “Partial membership is allowed in fuzzy set”**

The term **fuzzy** refers to things which are not clear or are vague. In the real world many times we encounter a situation when we can’t determine whether the state is true or false, their fuzzy logic provides a very valuable flexibility for reasoning. In this way, we can consider the inaccuracies and uncertainties of any situation.

In Boolean system truth value, 1.0 represents absolute truth value and 0.0 represents absolute false value. But in the fuzzy system, there is no logic for absolute truth and absolute false value. But in fuzzy logic, there is intermediate value too present which is partially true and partially false.



In other words, we can say that fuzzy logic is not logic that is fuzzy, but logic that is used to describe fuzziness. There can be numerous other examples like this with the help of which we can understand the concept of fuzzy logic.

**1g) Mention the role of fitness function in Genetic Algorithm.**

**Fitness Function** (also known as the **Evaluation Function**) evaluates how close a given solution is to the optimum solution of the desired problem. It determines how fit a solution is.

**Why we use Fitness Functions?**

In genetic algorithms, each solution is generally represented as a string of binary numbers, known as a **chromosome**. We have to test these solutions and come up with the best set of solutions to solve a given problem. Each solution, therefore, needs to be awarded a score, to indicate how close it came to meeting the overall specification of the desired solution. This score is generated by applying the **fitness function** to the test, or results obtained from the tested solution.

**1h) Differentiate between supervised and unsupervised learning?**

1. Supervised learning technique deals with the labelled data where the output data patterns are known to the system. As against, the unsupervised learning works with unlabeled data in which the output is just based on the collection of perceptions.
2. When it comes to the complexity the supervised learning method is less complex while unsupervised learning method is more complicated.
3. The supervised learning can also conduct offline analysis whereas unsupervised learning employs real-time analysis.
4. The outcome of the supervised learning technique is more accurate and reliable. In contrast, unsupervised learning generates moderate but reliable results.
5. Classification and regression are the types of problems solved under the supervised learning method. Conversely, unsupervised learning includes clustering and associative rule mining problems.

**1i) What do you mean by optimization function?**

Optimization is the process of finding the greatest or least value of a function for some constraint, which must be true regardless of the solution. This process is commonly used in computer science and physics, often called energy optimization. For a function *f*(x), called the objective function, that has a domain of real numbers of set *A*, the maximum optimal solution occurs where https://media.cheggcdn.com/study/e69/e697c3b2-868b-4d5d-ba8a-d2fe532c1dea/calc-26-eq-1.pngover set *A* and the minimum optimal solution occurs where https://media.cheggcdn.com/study/f74/f74de4af-8767-4221-866e-dc550b6d2f2a/calc-26-eq-2.png over set *A*. The three general ways of optimizing a function are to

1) find the absolute extrema of the function,

2) use the first derivative test, or

3) use the second derivative test.

1J) Briefly explain chemotaxis in bacterial foraging algorithm

A bacterium takes foraging decisions after considering two previous factors. The process, in which a bacterium moves by taking small steps while searching for nutrients, is called chemotaxis and key idea of BFOA is mimicking chemotactic movement of virtual bacteria in the problem search space.

Chemotaxis is achieved by swimming and tumbling. When a bacterium meets a favourable environment (rich in nutrients, and noxious free), it will continue swimming in the same direction. When it meets an unfavourable environment, it will tumble, i.e., change direction. Let S be the total number of bacteria in the population, and a bacterium position represents a candidate solution of the problem and information of the i-th bacterium with a dimensional vector represented as , i = 1, 2, ..., S. Suppose θi (j,k,l) represents the i-th bacterium at the j-th chemotactic, k-th reproductive, and l-th elimination and dispersal step. Then in computational chemotaxis, the movement of the bacterium may be represented by

θi (j+1, k, l)=θi (j, k, l)+C(i)Φ(j)

where C(i) is the size of the step taken in the random direction specified by the tumble (run length unit), and Φ(j) is in the random direction specified by the tumble.

2a) What is the necessity of activation function? List some commonly used activation function.

**Definition of activation function:-**Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to **introduce non-linearity** into the output of a neuron.

A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

**Explanation :-**  
We know, neural network has neurons that work in correspondence of *weight, bias* and their respective activation function. In a neural network, we would update the weights and biases of the neurons on the basis of the error at the output. This process is known as *back-propagation*. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases.

**VARIANTS OF ACTIVATION FUNCTION :-**

**1). Linear Function :-**

* **Equation :**Linear function has the equation similar to as of a straight line i.e. **y = ax**
* No matter how many layers we have, if all are linear in nature, the final activation function of last layer is nothing but just a linear function of the input of first layer.
* **Range :** -inf to +inf
* **Uses : Linear activation function** is used at just one place i.e. output layer.
* **Issues :**If we will differentiate linear function to bring non-linearity, result will no more depend on *input “x”* and function will become constant, it won’t introduce any ground-breaking behavior to our algorithm.

**For example :** Calculation of price of a house is a regression problem. House price may have any big/small value, so we can apply linear activation at output layer. Even in this case neural net must have any non-linear function at hidden layers.

**2). Sigmoid Function :-**

* It is a function which is plotted as **‘S’** shaped graph.
* **Equation :**  
  A = 1/(1 + e-x)
* **Nature :** Non-linear. Notice that X values lies between -2 to 2, Y values are very steep. This means, small changes in x would also bring about large changes in the value of Y.
* **Value Range :**0 to 1
* **Uses :**Usually used in output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to be ***1*** if value is greater than **0.5** and ***0*** otherwise.

**3). Tanh Function :-**The activation that works almost always better than sigmoid function is Tanh function also knows as Tangent Hyperbolic function. It’s actually mathematically shifted version of the sigmoid function. Both are similar and can be derived from each other.

**Equation :-**

**f(x) = tanh(x) = 2/(1 + e-2x) - 1**

**OR**

**tanh(x) = 2 \* sigmoid(2x) - 1**

* Value Range :- -1 to +1
* Nature :- non-linear
* Uses :- Usually used in hidden layers of a neural network as it’s values lies between -1 to 1 hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in centering the data by bringing mean close to 0. This makes learning for the next layer much easier.

**4).RELU :-**Stands for *Rectified linear unit*. It is the most widely used activation function. Chiefly implemented in *hidden layers* of Neural network.

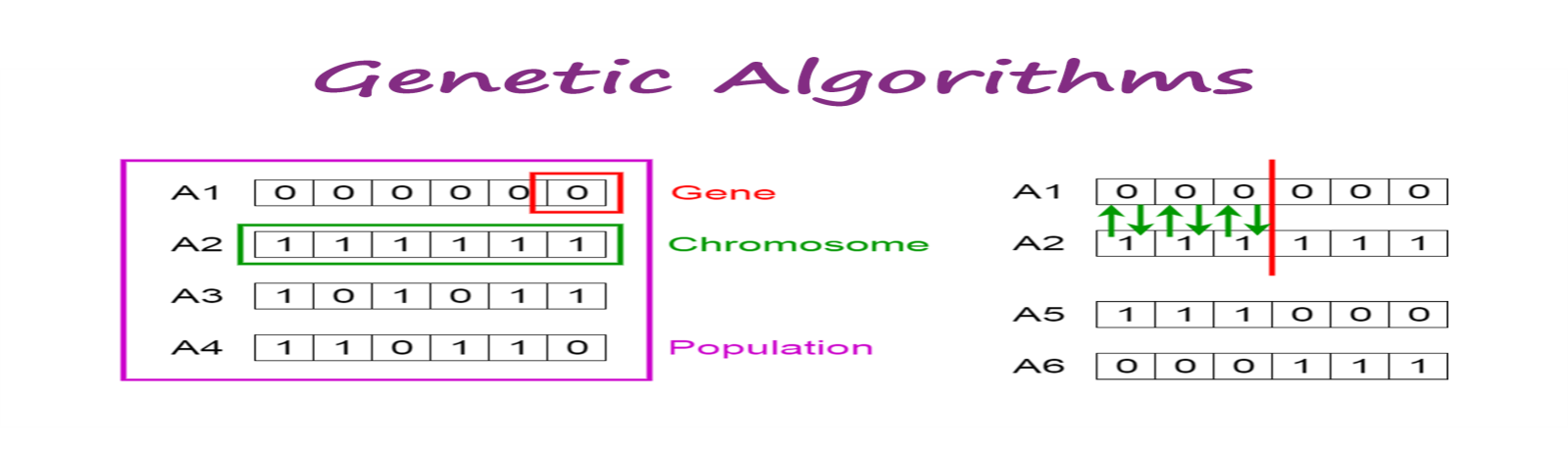
* **Equation :- *A(x) = max(0,x)***. It gives an output x if x is positive and 0 otherwise.
* **Value Range :-**[0, inf)
* **Nature :-**non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.
* **Uses :-**ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.

In simple words, RELU learns *much faster* than sigmoid and Tanh function.

**5). Softmax Function :-**The softmax function is also a type of sigmoid function but is handy when we are trying to handle classification problems.

* **Nature :-**non-linear
* **Uses :-**Usually used when trying to handle multiple classes. The softmax function would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of the outputs.
* **Ouput:-**The softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.

2b) Define the terms chromosome, fitness function, crossover and mutation is used in genetic Algorithms. Explain how genetic algorithm work.

A **genetic algorithm** is a search heuristic that is inspired by Charles Darwin’s theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation**Notion of Natural Selection**

The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation.

This notion can be applied for a search problem. We consider a set of solutions for a problem and select the set of best ones out of them.

Five phases are considered in a genetic algorithm.

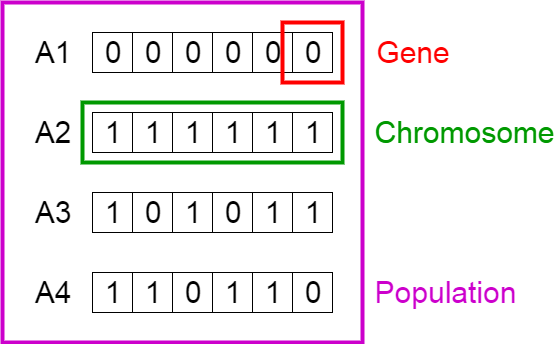
1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

**Initial Population**

The process begins with a set of individuals which is called a **Population**. Each individual is a solution to the problem you want to solve.

An individual is characterized by a set of parameters (variables) known as **Genes**. Genes are joined into a string to form a **Chromosome** (solution).

In a genetic algorithm, the set of genes of an individual is represented using a string, in terms of an alphabet. Usually, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome.



**Fitness Function**

The **fitness function** determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a **fitness score** to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

**Selection**

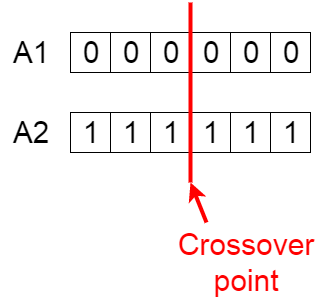
The idea of **selection** phase is to select the fittest individuals and let them pass their genes to the next generation.

Two pairs of individuals (**parents**) are selected based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction.

**Crossover**

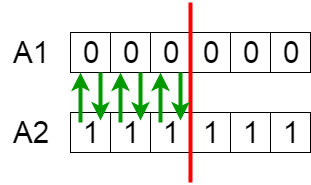
**Crossover** is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a **crossover point** is chosen at random from within the genes.

For example, consider the crossover point to be 3 as shown below.



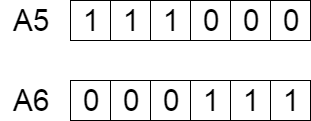
Crossover point

**Offspring** are created by exchanging the genes of parents among themselves until the crossover point is reached.



Exchanging genes among parents

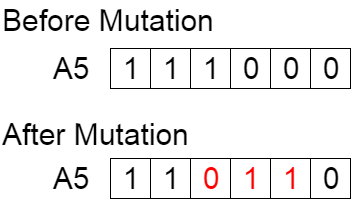
The new offspring are added to the population.



New offspring

**Mutation**

In certain new offspring formed, some of their genes can be subjected to a **mutation** with a low random probability. This implies that some of the bits in the bit string can be flipped.



Mutation: Before and After

Mutation occurs to maintain diversity within the population and prevent premature convergence.

**Termination**

The algorithm terminates if the population has converged (does not produce offspring which are significantly different from the previous generation). Then it is said that the genetic algorithm has provided a set of solutions to our problem.

**Comments**

The population has a fixed size. As new generations are formed, individuals with least fitness die, providing space for new offspring.

The sequence of phases is repeated to produce individuals in each new generation which are better than the previous generation.

**Psuedocode**

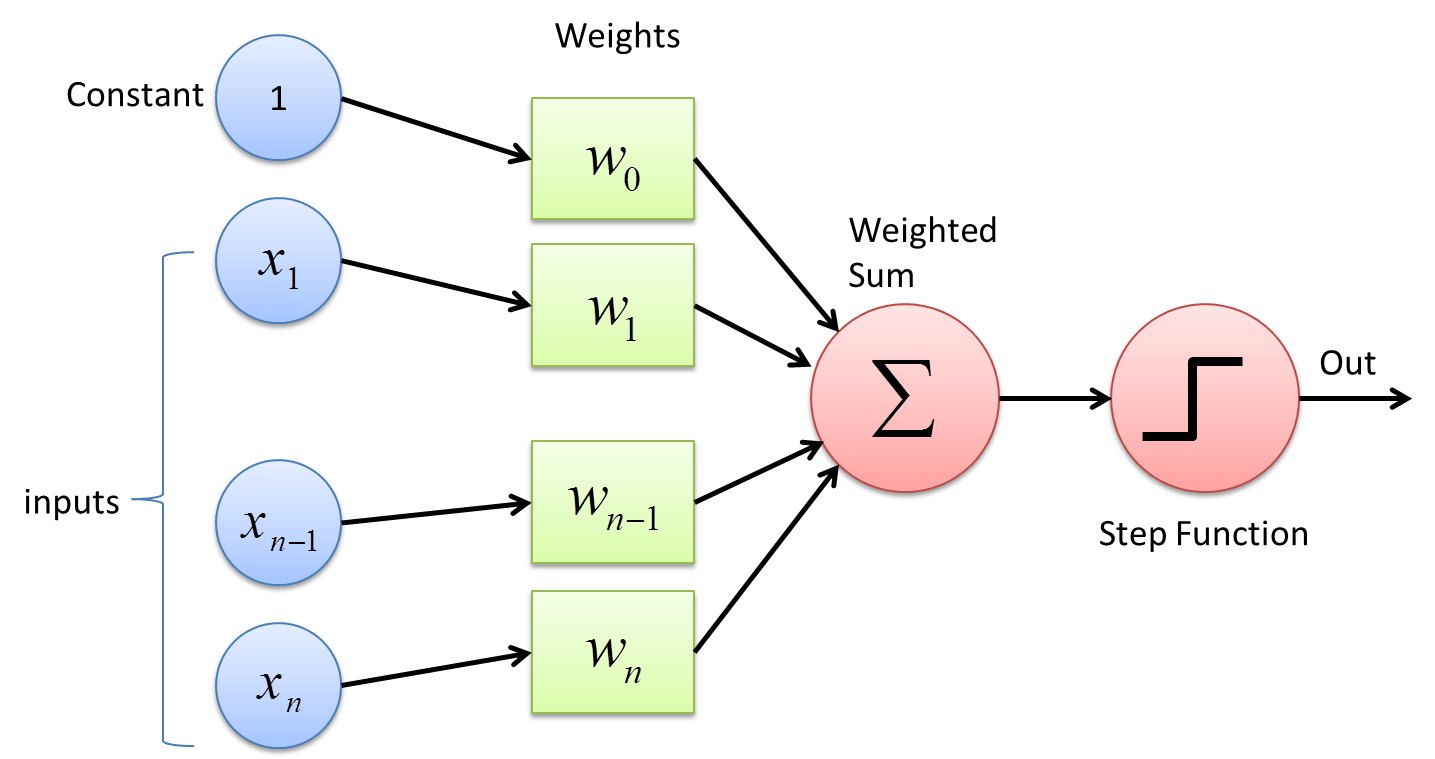
START  
Generate the initial population  
Compute fitness  
REPEAT  
 Selection  
 Crossover  
 Mutation  
 Compute fitness  
UNTIL population has converged  
STOP

2c) Train perceptron network for learning binary inputs and bipolar output or Gate function

**Perceptron**

Developed by Frank Rosenblatt by using McCulloch and Pitts model, perceptron is the basic operational unit of artificial neural networks. It employs supervised learning rule and is able to classify the data into two classes.

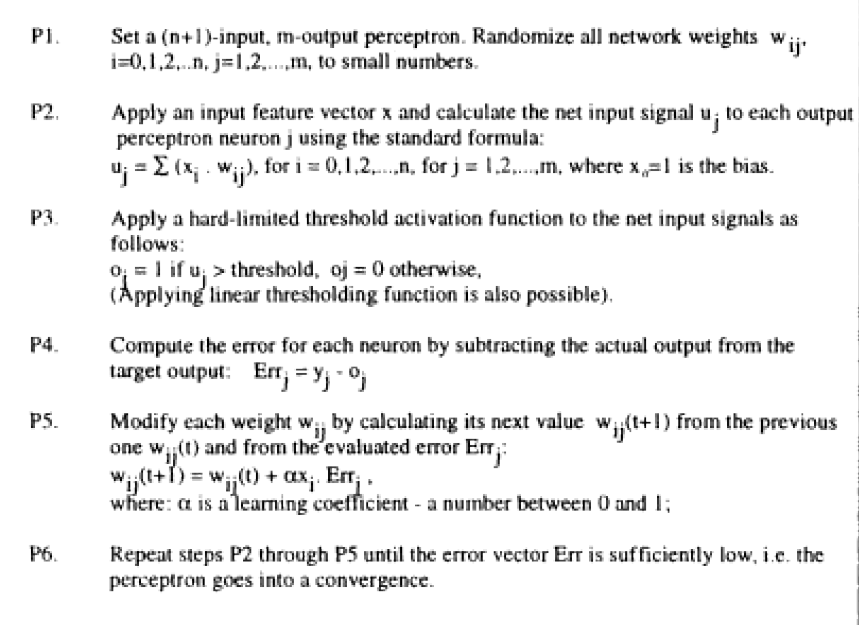
Operational characteristics of the perceptron: It consists of a single neuron with an arbitrary number of inputs along with adjustable weights, but the output of the neuron is 1 or 0 depending upon the threshold. It also consists of a bias whose weight is always 1. Following figure gives a schematic representation of the perceptron.



Perceptron thus has the following three basic elements −

* **Links** − It would have a set of connection links, which carries a weight including a bias always having weight 1.
* **Adder** − It adds the input after they are multiplied with their respective weights.
* **Activation function** − It limits the output of neuron. The most basic activation function is a Heaviside step function that has two possible outputs. This function returns 1, if the input is positive, and 0 for any negative input.

### Training Algorithm

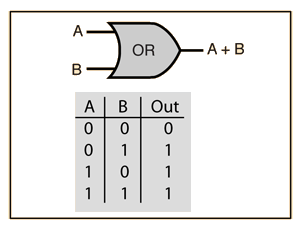
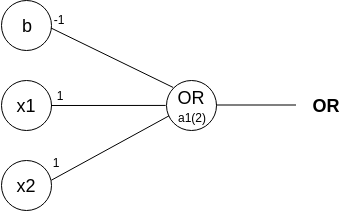
****

**How does it work**?

The perceptron works on these simple steps

1. All the inputs x are multiplied with their weights w. Let’s call it k.
2. Add all the multiplied values and call them Weighted Sum.
3. Apply that weighted sum to the correct Activation Function.

**OR Gate**

From the diagram, the OR gate is 0 only if both inputs are 0.

**Row 1**

* From w1x1+w2x2+b, initializing w1, w2, as 1 and b as -1, we get;

*x1(1)+x2(1)-1*

* Passing the first row of the OR logic table (x1=0, x2=0), we get;

*0+0–1 = -1*

* From the Perceptron rule, if Wx+b<0, then y`=0. Therefore, this row is correct.

**Row 2**

* Passing (x1=0 and x2=1), we get;

*0+1–1 = 0*

* From the Perceptron rule, if Wx+b >**=** 0, then y`=1. This row is again, correct (for both row 1, row 2 and 3).

**Row 4**

* Passing (x1=1 and x2=1), we get;

*1+1–1 = 1*

* Again, from the perceptron rule, this is still valid. Quite Easy!

Therefore, we can conclude that the model to achieve an OR gate, using the Perceptron algorithm is;

*x1+x2–1*

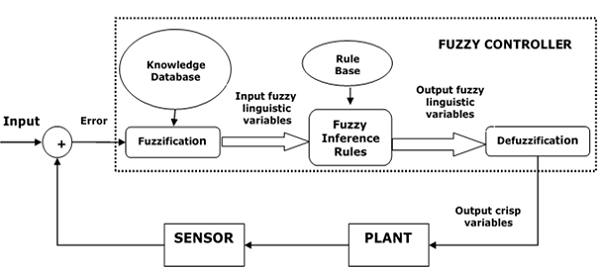
**2d) Draw a generalized flow chart for the design of FLC. Illustrate the formation rules and decision making logic with case study.**

A control system is an arrangement of physical components designed to alter another physical system so that this system exhibits certain desired characteristics. Following are some reasons of using Fuzzy Logic in Control Systems −

* While applying traditional control, one needs to know about the model and the objective function formulated in precise terms. This makes it very difficult to apply in many cases.
* By applying fuzzy logic for control we can utilize the human expertise and experience for designing a controller.
* The fuzzy control rules, basically the IF-THEN rules, can be best utilized in designing a controller.

**Architecture of Fuzzy Logic Control**

The following diagram shows the architecture of Fuzzy Logic Control (FLC).



**Major Components of FLC**

Followings are the major components of the FLC as shown in the above figure −

* Fuzzifier − The role of fuzzifier is to convert the crisp input values into fuzzy values.
* Fuzzy Knowledge Base − It stores the knowledge about all the input-output fuzzy relationships. It also has the membership function which defines the input variables to the fuzzy rule base and the output variables to the plant under control.
* Fuzzy Rule Base − It stores the knowledge about the operation of the process of domain.
* Inference Engine − It acts as a kernel of any FLC. Basically it simulates human decisions by performing approximate reasoning.
* Defuzzifier − The role of defuzzifier is to convert the fuzzy values into crisp values getting from fuzzy inference engine.

**Steps in Designing FLC**

Following are the steps involved in designing FLC −

* Identification of variables − Here, the input, output and state variables must be identified of the plant which is under consideration.
* Fuzzy subset configuration − The universe of information is divided into number of fuzzy subsets and each subset is assigned a linguistic label. Always make sure that these fuzzy subsets include all the elements of universe.
* Obtaining membership function − Now obtain the membership function for each fuzzy subset that we get in the above step.
* Fuzzy rule base configuration − Now formulate the fuzzy rule base by assigning relationship between fuzzy input and output.
* Fuzzification − The fuzzification process is initiated in this step.
* Combining fuzzy outputs − By applying fuzzy approximate reasoning, locate the fuzzy output and merge them.
* Defuzzification − Finally, initiate defuzzification process to form a crisp output.

**Advantages of Fuzzy Logic Control**

Let us now discuss the advantages of Fuzzy Logic Control.

* Cheaper − Developing a FLC is comparatively cheaper than developing model based or other controller in terms of performance.
* Robust − FLCs are more robust than PID controllers because of their capability to cover a huge range of operating conditions.
* Customizable − FLCs are customizable.
* Emulate human deductive thinking − Basically FLC is designed to emulate human deductive thinking, the process people use to infer conclusion from what they know.
* Reliability − FLC is more reliable than conventional control system.
* Efficiency − Fuzzy logic provides more efficiency when applied in control system.

2e) How will you apply NN optimization to

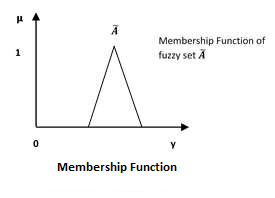
Graph bipartition problem

Linear programming problem

Travelling sales man problems

2f) Define membership function and explain importance in fuzzy logic.

Fuzzy logic is not logic that is fuzzy but logic that is used to describe fuzziness. This fuzziness is best characterized by its membership function. In other words, we can say that membership function represents the degree of truth in fuzzy logic.



Following are a few important points relating to the membership function −

* Membership functions were first introduced in 1965 by Lofti A. Zadeh in his first research paper “fuzzy sets”.
* Membership functions characterize fuzziness (i.e., all the information in fuzzy set), whether the elements in fuzzy sets are discrete or continuous.
* Membership functions can be defined as a technique to solve practical problems by experience rather than knowledge.
* Membership functions are represented by graphical forms.
* Rules for defining fuzziness are fuzzy too.

**Mathematical Notation**

Fuzzy set *Ã* in the universe of information *U* can be defined as a set of ordered pairs and it can be represented mathematically as −



Here **μA˜(∙)** = membership function of **A˜**; this assumes values in the range from 0 to 1, i.e., **μA˜(∙)∈[0,1]**.

The membership function **μA˜(∙)** maps **U** to the membership space **M**.

The dot **(∙)** in the membership function described above, represents the element in a fuzzy set; whether it is discrete or continuous.

**Features of Membership Functions**

Different features of Membership Functions.

**Core**

For any fuzzy set **A~**, the core of a membership function is that region of universe that is characterize by full membership in the set. Hence, core consists of all those elements **y** of the universe of information such that,

**μ A~ (y)=1**

**Support**

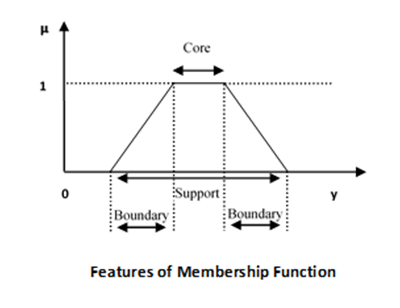
For any fuzzy set **A~**, the support of a membership function is the region of universe that is characterize by a nonzero membership in the set. Hence core consists of all those elements **y** of the universe of information such that,

****

**Boundary**

For any fuzzy set **A~**, the boundary of a membership function is the region of universe that is characterized by a nonzero but incomplete membership in the set. Hence, core consists of all those elements **y** of the universe of information such that,





2h) Write the algorithm for steepest ascent hill climbing and explain how this technique helping to find out an optimized solution for problems that can not be solved easily.

Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbor has a higher value.

Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is Traveling-salesman Problem in which we need to minimize the distance traveled by the salesman.

The steepest-Ascent algorithm is a variation of simple hill climbing algorithm. This algorithm examines all the neighboring nodes of the current state and selects one neighbor node which is closest to the goal state. This algorithm consumes more time as it searches for multiple neighbors

### Algorithm for Steepest-Ascent hill climbing:

* **Step 1:** Evaluate the initial state, if it is goal state then return success and stop, else make current state as initial state.
* **Step 2:** Loop until a solution is found or the current state does not change.
  1. Let SUCC be a state such that any successor of the current state will be better than it.
  2. For each operator that applies to the current state:
     1. Apply the new operator and generate a new state.
     2. Evaluate the new state.
     3. If it is goal state, then return it and quit, else compare it to the SUCC.
     4. If it is better than SUCC, then set new state as SUCC.
     5. If the SUCC is better than the current state, then set current state to SUCC.
* **Step 5:** Exit.

Hill climbing cannot reach the optimal/best state(global maximum) if it enters any of the following regions :

1. **Local maximum :**At a local maximum all neighboring states have a values which is worse than the current state. Since hill-climbing uses a greedy approach, it will not move to the worse state and terminate itself. The process will end even though a better solution may exist.  
   **To overcome local maximum problem :**Utilize backtracking technique. Maintain a list of visited states. If the search reaches an undesirable state, it can backtrack to the previous configuration and explore a new path.
2. **Plateau :**On plateau all neighbors have same value . Hence, it is not possible to select the best direction.

**To overcome plateaus :** Make a big jump. Randomly select a state far away from the current state. Chances are that we will land at a non-plateau region

1. **Ridge :**Any point on a ridge can look like peak because movement in all possible directions is downward. Hence the algorithm stops when it reaches this state.  
   **To overcome Ridge :** In this kind of obstacle, use two or more rules before testing. It implies moving in several directions at once.

2i) Using Gauss-Jordan method, solve the system equation.

10x+y+z=12

X+10y+z=12

X+y+10z=12

2j) List out12 no of Engineering problems, where optimization can be applied to solve the engineering problem.

**2k) Explain in brief about the necessity of Defuzzification process and flow a fuzzy relationship converted to crisp relationship using ‘lambada-cut process’**

**Defuzzification**

It may be defined as the process of reducing a fuzzy set into a crisp set or to convert a fuzzy member into a crisp member.

Fuzzification process involves conversion from crisp quantities to fuzzy quantities. In a number of engineering applications, it is necessary to defuzzify the result or rather “fuzzy result” so that it must be converted to crisp result. Mathematically, the process of Defuzzification is also called “rounding it off”.

Lmabda-cut method is applicable to derive crisp value of a fuzzy set or relation.

Thus Lambda-cut method for fuzzy set

Lambda-cut method for fuzzy relation

In many literature, Lambda-cut method is also alternatively termed as Alph-cut method.

In this method a fuzzy set A is transformed into a crisp set Aλ for a given value of λ (0 ≤ λ ≤ 1) 2 In other-words, Aλ = {x|µA(x) ≥ λ} 3

That is, the value of Lambda-cut set Aλ is x, when the membership value corresponding to x is greater than or equal to the specified λ. 4 This Lambda-cut set Aλ is also called alpha-cut set

A1 = {(x1, 0.9),(x2, 0.5),(x3, 0.2),(x4, 0.3)}

Then A0.6 = {(x1, 1),(x2, 0),(x3, 0),(x4, 0)} = {x1}

and

A2 = {(x1, 0.1),(x2, 0.5),(x3, 0.8),(x4, 0.7)}

A0.2 = {(x1, 0),(x2, 1),(x3, 1),(x4, 1)} = {x2, x3, x4}

**2l) Briefly explain about different linear programming methods**

**5)** The process by which a Multi Layer Perceptron learns is called the Back-proagation algorithm.

**Backward Propagation of Errors,**often abbreviated as BackProp is one of the several ways in which an artificial neural network (ANN) can be trained. It is a supervised training scheme, which means, it learns from labeled training data (there is a supervisor, to guide its learning).

**BackProp Algorithm:**  
Initially all the edge weights are randomly assigned. For every input in the training dataset, the ANN is activated and its output is observed. This output is compared with the desired output that we already know, and the error is “propagated” back to the previous layer. This error is noted and the weights are “adjusted” accordingly. This process is repeated until the output error is below a predetermined threshold.

Once the above algorithm terminates, we have a “learned” ANN which, we consider is ready to work with “new” inputs. This ANN is said to have learned from several examples (labeled data) and from its mistakes (error propagation).

The Multi Layer Perceptron shown in below Figure has two nodes in the input layer (apart from the Bias node) which take the inputs ‘Hours Studied’ and ‘Mid Term Marks’. It also has a hidden layer with two nodes (apart from the Bias node). The output layer has two nodes as well – the upper node outputs the probability of ‘Pass’ while the lower node outputs the probability of ‘Fail’.

In classification tasks, we generally use a Softmax function as the Activation Function in the Output layer of the Multi Layer Perceptron to ensure that the outputs are probabilities and they add up to 1. The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sum to one. So, in this case, Probability (Pass) + Probability (Fail) = 1

**Step 1: Forward Propagation**

All weights in the network are randomly assigned. Lets consider the hidden layer node marked **V** in above Figure  below. Assume the weights of the connections from the inputs to that node are w1, w2 and w3 (as shown).

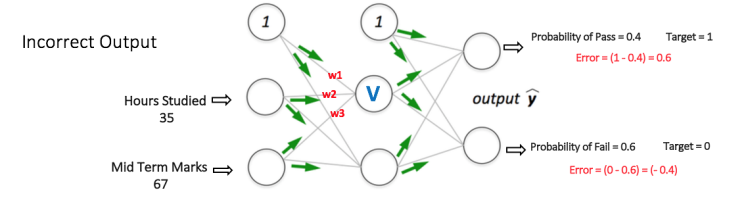
The network then takes the first training example as input (we know that for inputs 35 and 67, the probability of Pass is 1).

* Input to the network = [35, 67]
* Desired output from the network (target) = [1, 0]

Then output V from the node in consideration can be calculated as below (***f***is an activation function such as sigmoid): V = ***f***(1\*w1 + 35\*w2 + 67\*w3)

Similarly, outputs from the other node in the hidden layer is also calculated. The outputs of the two nodes in the hidden layer act as inputs to the two nodes in the output layer. This enables us to calculate output probabilities from the two nodes in output layer.

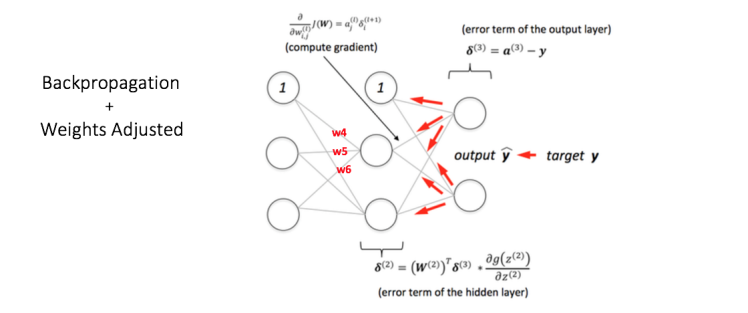
Suppose the output probabilities from the two nodes in the output layer are 0.4 and 0.6 respectively (since the weights are randomly assigned, outputs will also be random). We can see that the calculated probabilities (0.4 and 0.6) are very far from the desired probabilities (1 and 0 respectively), hence the network in Figure 5 is said to have an ‘Incorrect Output’.



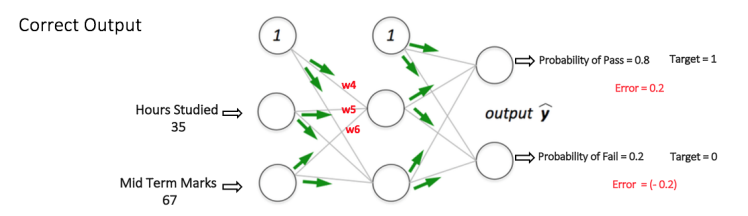
**Step 2: Back Propagation and Weight Updation**

We calculate the total error at the output nodes and propagate these errors back through the network using Back propagation to calculate the *gradients*. Then we use an optimization method such as *Gradient Descent* to ‘adjust’ **all** weights in the network with an aim of reducing the error at the output layer. This is shown in the Figure 6 below (ignore the mathematical equations in the figure for now).

Suppose that the new weights associated with the node in consideration are w4, w5 and w6 (after Backpropagation and adjusting weights).

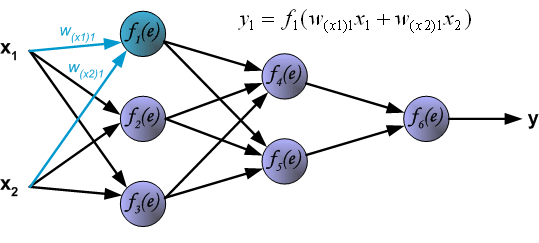


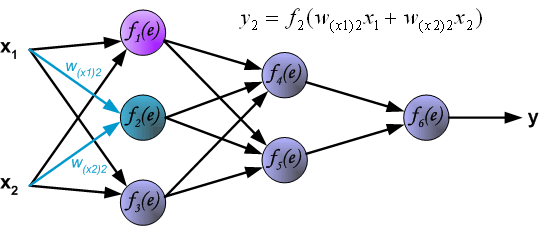
If we now input the same example to the network again, the network should perform better than before since the weights have now been adjusted to minimize the error in prediction. As shown in Figure 7, the errors at the output nodes now reduce to [0.2, -0.2] as compared to [0.6, -0.4] earlier. This means that our network has learnt to correctly classify our first training example.

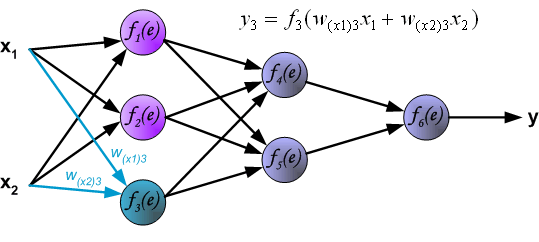


We repeat this process with all other training examples in our dataset. Then, our network is said to have *learnt*those examples.

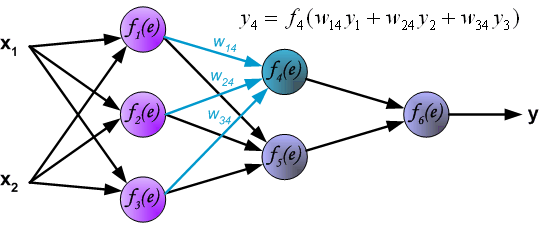
If we now want to predict whether a student studying 25 hours and having 70 marks in the mid term will pass the final term, we go through the forward propagation step and find the output probabilities for Pass and Fail.

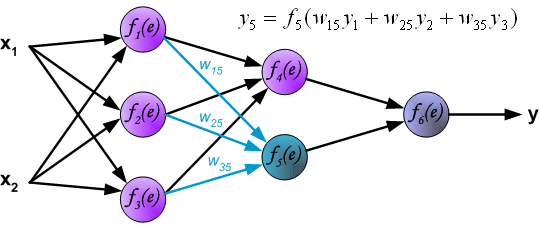




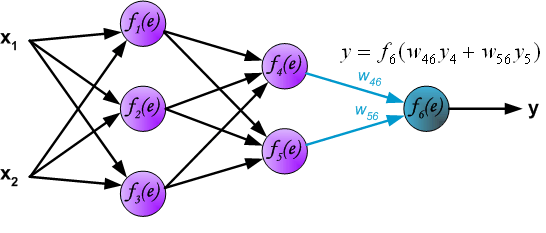


Propagation of signals through the hidden layer. Symbols *wmn* represent weights of connections between output of neuron *m* and input of neuron *n* in the next layer.

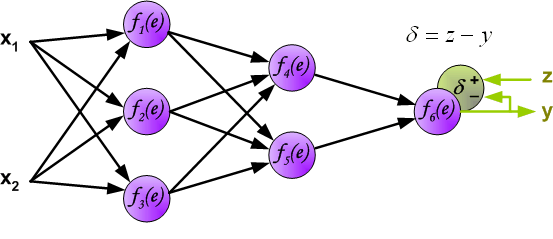




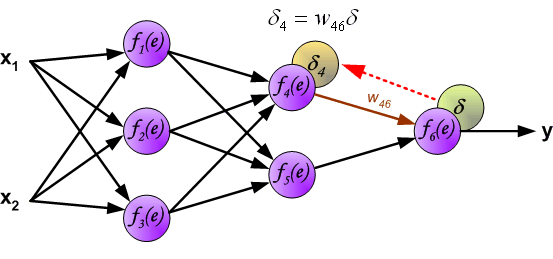
Propagation of signals through the output layer.

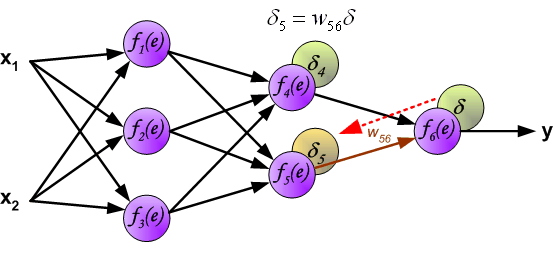


In the next algorithm step the output signal of the network *y* is compared with the desired output value (the target), which is found in training data set. The difference is called error signal *d* of output layer neuron.

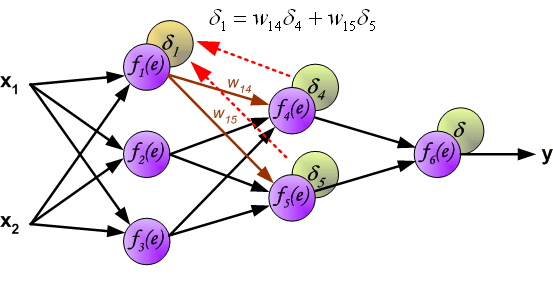


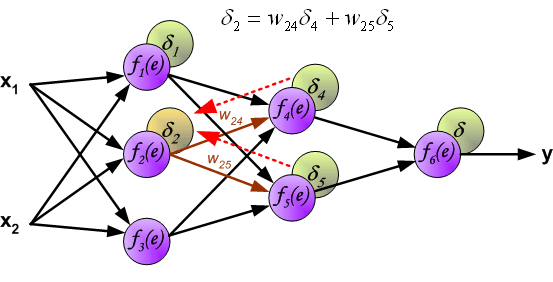
It is impossible to compute error signal for internal neurons directly, because output values of these neurons are unknown. For many years the effective method for training multiplayer networks has been unknown. Only in the middle eighties the backpropagation algorithm has been worked out. The idea is to propagate error signal *d* (computed in single teaching step) back to all neurons, which output signals were input for discussed neuron.





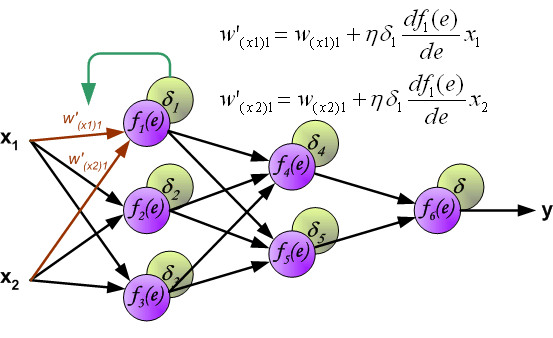
The weights' coefficients *wmn* used to propagate errors back are equal to this used during computing output value. Only the direction of data flow is changed (signals are propagated from output to inputs one after the other). This technique is used for all network layers. If propagated errors came from few neurons they are added. The illustration is below.

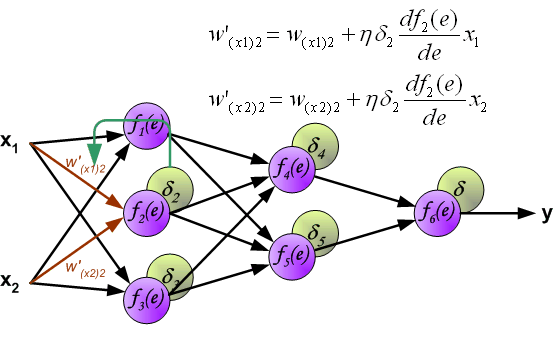


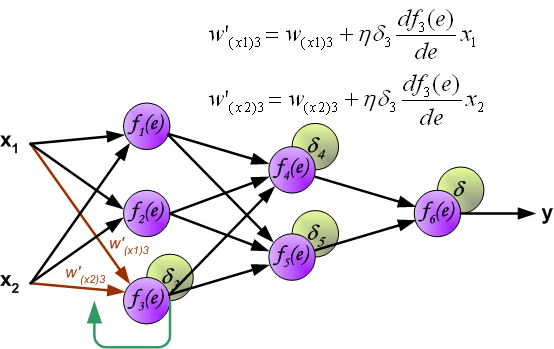


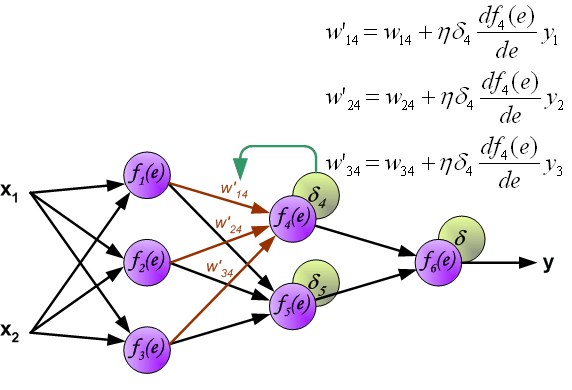


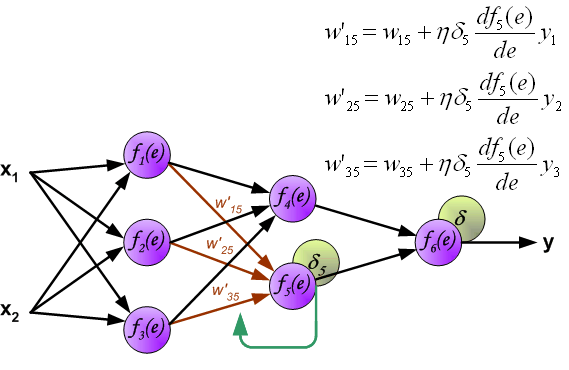
When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified. In formulas below *df(e)/de* represents derivative of neuron activation function (which weights are modified).

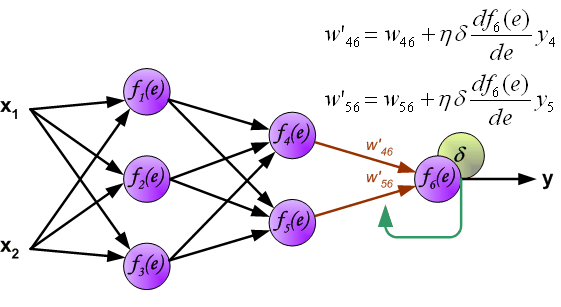












Coefficient *h* affects network teaching speed. There are a few techniques to select this parameter. The first method is to start teaching process with large value of the parameter. While weights coefficients are being established the parameter is being decreased gradually. The second, more complicated, method starts teaching with small parameter value. During the teaching process the parameter is being increased when the teaching is advanced and then decreased again in the final stage. Starting teaching process with low parameter value enables to determine weights coefficients signs.

