1. Describe the structure of an artificial neuron. How is it similar to a biological neuron? What are its main components?

**Ans: An artificial neuron is a connection point in an artificial neural network. Artificial neural networks, like the human body's biological neural network, have a layered architecture and each network node (connection point) has the capability to process input and forward output to other nodes in the network. Biological neurons have only provided an inspiration to their artificial counterparts, but they are in no way direct copies with similar potential.**

1. What are the different types of activation functions popularly used? Explain each of them.

**Ans: 1. Binary Step Function**

**The first thing that comes to our mind when we have an activation function would be a threshold based classifier i.e. whether or not the neuron should be activated based on the value from the linear transformation.**

**2. Linear Function**

**We saw the problem with the step function, the gradient of the function became zero. This is because there is no component of x in the binary step function. Instead of a binary function, we can use a linear function.**

**3. Sigmoid**

**The next activation function that we are going to look at is the Sigmoid function. It is one of the most widely used non-linear activation function. Sigmoid transforms the values between the range 0 and 1.**

**4. Tanh**

**The tanh function is very similar to the sigmoid function. The only difference is that it is symmetric around the origin. The range of values in this case is from -1 to 1. Thus the inputs to the next layers will not always be of the same sign.**

**5. ReLU**

**The ReLU function is another non-linear activation function that has gained popularity in the deep learning domain. ReLU stands for Rectified Linear Unit. The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time.**

**6. Leaky ReLU**

**Leaky ReLU function is nothing but an improved version of the ReLU function. As we saw that for the ReLU function, the gradient is 0 for x<0, which would deactivate the neurons in that region.**

**7. Parameterised ReLU**

**This is another variant of ReLU that aims to solve the problem of gradient’s becoming zero for the left half of the axis. The parameterised ReLU, as the name suggests, introduces a new parameter as a slope of the negative part of the function.**

**8. Exponential Linear Unit**

**Exponential Linear Unit or ELU for short is also a variant of Rectiufied Linear Unit (ReLU) that modifies the slope of the negative part of the function. Unlike the leaky relu and parametric ReLU functions, instead of a straight line, ELU uses a log curve for defning the negatice values.**

**9.Swish**

**Swish is a lesser known activation function which was discovered by researchers at Google. Swish is as computationally efficient as ReLU and shows better performance than ReLU on deeper models. The values for swish ranges from negative infinity to infinity.**

**10. Softmax**

**Softmax function is often described as a combination of multiple sigmoids. We know that sigmoid returns values between 0 and 1, which can be treated as probabilities of a data point belonging to a particular class. Thus sigmoid is widely used for binary classification problems.**

* 1. Explain, in details, Rosenblatt’s perceptron model. How can a set of data be classified using a simple perceptron?

**Rosenblatt’s major achievement has been to show that, by relaxing some of the MCP’s rules (namely the absolute inhibition, the equal contribution of all inputs as well as their integer nature), artificial neurons could actually learn from data. More importantly, he came up with a supervised learning algorithm for this modified MCP neuron model that enabled the artificial neuron to figure out the correct weights directly from training data by itself. The Perceptron is a linear classification algorithm. This means that it learns a decision boundary that separates two classes using a line (called a hyperplane) in the feature space. As such, it is appropriate for those problems where the classes can be separated well by a line or linear model, referred to as linearly separable.**

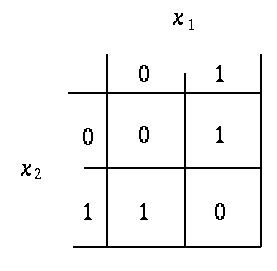
1. Explain the basic structure of a multi-layer perceptron. Explain how it can solve the XOR problem.

**Ans: A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function.**

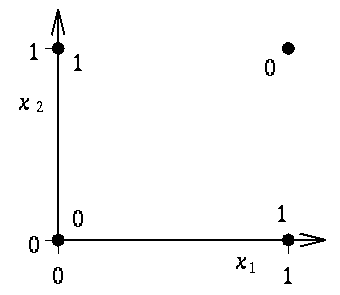
**The XOR problem**

**This is the simplest problem that can not be solved by a perceptron.**

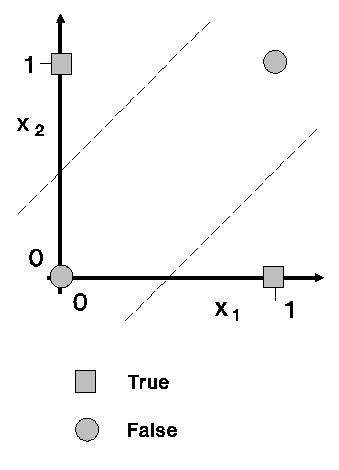
**For two inputs x1 and x2, the output is the exclusive OR of the inputs.**

****

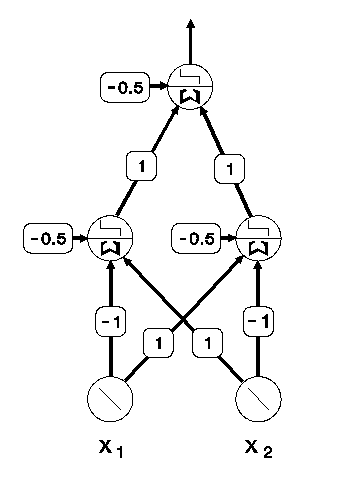
**The pattern space for this problem looks like this:**

****

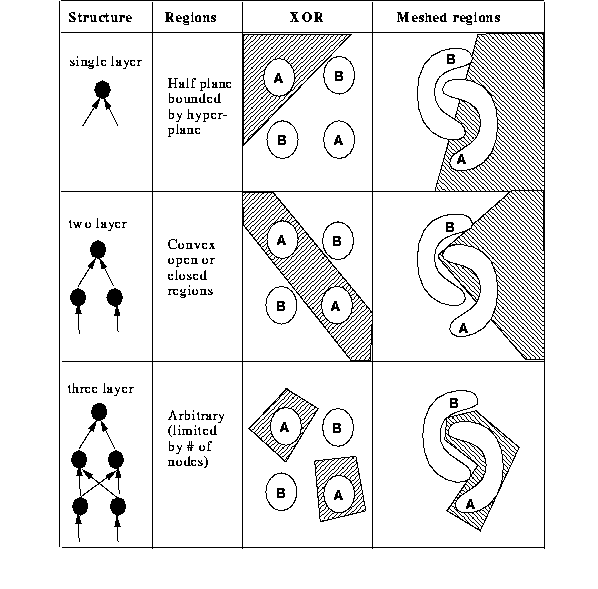
**This cannot be solved using a single line, the solution uses two lines:**

****

**A two layer Multi-Layer Perceptron to solve this problem looks like this:**

****

**The shape of regions in pattern space that can be separated by a Multi-Layer Perceptron is shown in the following table.**

****

**We can see that a three layer MLP can learn arbitrary areas while a two layer MLP can learn convex regions. (if you can draw a line from any point in the region to any other in the region and the line passes out of the region then that region is not convex).**

1. What is artificial neural network (ANN)? Explain some of the salient highlights in the different architectural options for ANN.

**Ans: An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the brain. ANNs, like people, learn by examples. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning largely involves adjustments to the synaptic connections that exist between the neurons. The model of Artificial neural network can be specified by three entities:**

* **Interconnections**
* **Activation functions**
* **Learning rules**

1. Explain the learning process of an ANN. Explain, with example, the challenge in assigning synaptic weights for the interconnection between neurons? How can this challenge be addressed?

**Ans: Learning process in ANN mainly depends on four factors, they are:**

* **The number of layers in the network (Single-layered or multi-layered)**
* **Direction of signal flow (Feedforward or recurrent)**
* **Number of nodes in layers: The number of node in the input layer is equal to the number of features of the input data set. The number of output nodes will depend on possible outcomes i.e. the number of classes in case of supervised learning. But the number of layers in the hidden layer is to be chosen by the user. A larger number of nodes in the hidden layer, higher the performance but too many nodes may result in overfitting as well as increased computational expense.**
* **Weight of Interconnected Nodes: Deciding the value of weights attached with each interconnection between each neuron so that a specific learning problem can be solved correctly is quite a difficult problem by itself. Take an example to understand the problem. Take the example of a Multi-layered Feed-Forward Network, we have to train an ANN model using some data, so that it can classify a new data set, say p\_5(3,-2). Say we have deduced that p\_1=(5,2) and p\_2 = (-1,12) belonging to class C1 while p\_3=(3,-5) and p\_4 = (-2,-1) belonging to class C2. We assume the values of synaptic weights w\_0,w\_1,w\_2 as -2, 1/2 and 1/4 respectively. But we will NOT get these weight values for every learning problem. For solving a learning problem with ANN, we can start with a set of values for synaptic weights and keep changing those in multiple iterations. The stopping criterion may be the rate of misclassification < 1% or the maximum numbers of iterations should be less than 25(a threshold value). There may be another problem that, the rate of misclassification may not reduce progressively.**

**So, we can summarize the learning process in ANN as the combination of – deciding the number of hidden layers, the number of nodes in each of the hidden layers, the direction of signal flow, deciding the connection weight.**

1. Explain, in details, the backpropagation algorithm. What are the limitations of this algorithm?

**Ans: Backpropagation is the essence of neural network training. It is the method of fine-tuning the weights of a neural network based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and make the model reliable by increasing its generalization.**

**Backpropagation in neural network is a short form for “backward propagation of errors.” It is a standard method of training artificial neural networks. This method helps calculate the gradient of a loss function with respect to all the weights in the network.**

**DISADVANTAGES OF BACK PROPAGATION ALGORITHM:**

**Though the advantages of backpropagation outnumber its disadvantages, it is still imperative to highlight these limitations. Therefore, here are the limitations of back propagation algorithms.**

* **It relies on input to perform on a specific problem.**
* **Sensitive to complex/noisy data.**
* **It needs the derivatives of activation functions for the network design time.**

1. Describe, in details, the process of adjusting the interconnection weights in a multi-layer neural network.

**Ans: One main part of the algorithm is adjusting the interconnection weights. This is done using a technique termed as Gradient Descent. In simple words, the algorithm calculates the partial derivative of the activation function by each interconnection weight to identify the ‘gradient’ or extent of change of the weight required to minimize the cost function.**

1. What are the steps in the backpropagation algorithm? Why a multi-layer neural network is required?

**Ans: last time we saw that the delta rule can be used to train a perceptron. When training the MLP, the error (delta) must be propagated back through the layers. This is called error back-propagation. Or just backpropagation.**

**The following procedure can be used to train a backpropagation network.**

**t is the target**

**units[l] is the number of units in layer l**

**n[l][i] is unit i in layer l**

**n[l][i].output is the output**

**n[l][i].delta is the delta**

**n[l][i].weight[j] is weight j**

**ek is the learning constant**

**adapt() {**

**int i,j,k,l;**

**for(l=layers-1;l>=0;l--)**

**for(i=0;i<units[l];i++)**

**if(l==layers-1)**

**n[l][i].delta=**

**ek\*n[l][i].output\***

**(1.0-n[l][i].output)\***

**(t[i]-n[l][i].output);**

**else {**

**n[l][i].delta=0.0;**

**for(k=0;k<units[l];k++)**

**n[l][i].delta+=**

**n[l+1][k].delta\***

**n[l+1][k].weight[i];**

**n[l][i].delta=n[l][i].delta\***

**ek\*n[l][i].output\***

**(1.0-n[l][i].output);**

**}**

**for(l=layers-1;l>=1;l--)**

**for(i=0;i<units[l];i++)**

**for(j=0;j<weights;j++)**

**n[l][i].weight[j]+=**

**n[l-1][j].output\***

**n[l][i].delta;**

**for(i=0;i<units[0];i++)**

**for(j=0;j<weights;j++)**

**n[0][i].weight[j]+=**

**input[j]\*n[0][i].delta;**

**}**

**When this algorithm is applied to the XOR we get the following output.**

**iteration no 0, inputs 0 1, target 1, output 0.477995**

**iteration no 20, inputs 1 0, target 1, output 0.447816**

**iteration no 40, inputs 0 0, target 0, output 0.450292**

**iteration no 60, inputs 1 0, target 1, output 0.549096**

**iteration no 80, inputs 0 0, target 0, output 0.460706**

**iteration no 100, inputs 1 0, target 1, output 0.507636**

**iteration no 120, inputs 0 1, target 1, output 0.571619**

**iteration no 140, inputs 0 0, target 0, output 0.451493**

**iteration no 160, inputs 0 1, target 1, output 0.570574**

**iteration no 180, inputs 1 0, target 1, output 0.575979**

**iteration no 200, inputs 0 1, target 1, output 0.744079**

**iteration no 220, inputs 1 1, target 0, output 0.233541**

**iteration no 240, inputs 0 1, target 1, output 0.755600**

**iteration no 260, inputs 1 1, target 0, output 0.185273**

**iteration no 280, inputs 0 1, target 1, output 0.788309**

**iteration no 300, inputs 1 1, target 0, output 0.167068**

**iteration no 320, inputs 0 0, target 0, output 0.123461**

**iteration no 340, inputs 1 1, target 0, output 0.132892**

**iteration no 360, inputs 1 1, target 0, output 0.133583**

**iteration no 380, inputs 1 1, target 0, output 0.116641**

**iteration no 400, inputs 0 0, target 0, output 0.088269**

**iteration no 420, inputs 1 0, target 1, output 0.861810**

**iteration no 440, inputs 1 1, target 0, output 0.102406**

**iteration no 460, inputs 0 0, target 0, output 0.080179**

**iteration no 480, inputs 0 0, target 0, output 0.075584**

**iteration no 500, inputs 1 0, target 1, output 0.884442**

**iteration no 520, inputs 1 0, target 1, output 0.892789**

**iteration no 540, inputs 0 1, target 1, output 0.923969**

**iteration no 560, inputs 0 0, target 0, output 0.064146**

**iteration no 580, inputs 1 1, target 0, output 0.071938**

**iteration no 600, inputs 1 1, target 0, output 0.075764**

**iteration no 620, inputs 1 1, target 0, output 0.074536**

**iteration no 640, inputs 1 1, target 0, output 0.069014**

**iteration no 660, inputs 1 1, target 0, output 0.066534**

**iteration no 680, inputs 1 0, target 1, output 0.918422**

**iteration no 700, inputs 1 0, target 1, output 0.924860**

**iteration no 720, inputs 1 1, target 0, output 0.065864**

**iteration no 740, inputs 1 1, target 0, output 0.052634**

**iteration no 760, inputs 1 0, target 1, output 0.927081**

**iteration no 780, inputs 0 0, target 0, output 0.050964**

**iteration no 800, inputs 0 1, target 1, output 0.948869**

**iteration no 820, inputs 0 0, target 0, output 0.049082**

**iteration no 840, inputs 0 0, target 0, output 0.048074**

**iteration no 860, inputs 1 1, target 0, output 0.057916**

**iteration no 880, inputs 1 1, target 0, output 0.056088**

**iteration no 900, inputs 0 1, target 1, output 0.954659**

**iteration no 920, inputs 1 1, target 0, output 0.057337**

**iteration no 940, inputs 1 0, target 1, output 0.944243**

**iteration no 960, inputs 0 0, target 0, output 0.045653**

**iteration no 980, inputs 1 0, target 1, output 0.946199**

1. Write short notes on:
   * + 1. Artificial neuron **: ANN learning is robust to errors in the training data and has been successfully applied for learning real-valued, discrete-valued, and vector-valued functions containing problems such as interpreting visual scenes, speech recognition, and learning robot control strategies.**
       2. Multi-layer perceptron **:** **The perceptron can only learn simple problems. It can place a hyperplane in pattern space and move the plane until the error is reduced. Unfortunately this is only useful if the problem is linearly separable.**
       3. Deep learning **: Deep learning is a subset of machine learning (ML), which is itself a subset of artificial intelligence (AI). The concept of AI has been around since the 1950s, with the goal of making computers able to think and reason in a way similar to humans. As part of making machines able to think, ML is focused on how to make them learn without being explicitly programmed. Deep learning goes beyond ML by creating more complex hierarchical models that are meant to mimic how humans learn new information.**
       4. Learning rate **: The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process.**
2. Write the difference between:-
   * + 1. Activation function vs threshold function

**Ans: To put it simply, activation functions are mathematical equations that determine the output of neural networks. They basically decide to deactivate neurons or activate them to get the desired output thus the name, activation functions.**

**In a neural network, input data points(x) which are numerical values are fed into neurons. Each and every neuron has a weight(w) which will be multiplied by the inputs and output a certain value which will again be fed into the neurons in the next layer. Activation functions come into the play as mathematical gates in between this process and decide whether the output of a certain neuron is on or off.**

**A threshold function is a Boolean function that determines whether a value equality of its inputs exceeded a certain threshold. A device that implements such logic is known as a threshold gate.**

* + - 1. Step function vs sigmoid function

**Ans: Binary step function is a threshold-based activation function which means after a certain threshold neuron is activated and below the said threshold neuron is deactivated. In the above graph, the threshold is zero. This activation function can be used in binary classifications as the name suggests, however it cannot be used in a situation where you have multiple classes to deal with.**

**Sigmoid function (also known as logistic function) takes a probabilistic approach and the output ranges between 0–1. It normalizes the output of each neuron. However, Sigmoid function makes almost no change in the prediction for very high or very low inputs which ultimately results in neural network refusing to learn further, this problem is known as the vanishing gradient.**

* + - 1. Single layer vs multi-layer perceptron

**Ans: A single layer perceptron (SLP) is a feed-forward network based on a threshold transfer function. SLP is the simplest type of artificial neural networks and can only classify linearly separable cases with a binary target (1 , 0).**

**A Multi Layer Perceptron (MLP) contains one or more hidden layers (apart from one input and one output layer). While a single layer perceptron can only learn linear functions, a multi layer perceptron can also learn non - linear functions**