Session 1 : Structure of Neural Networks

Artificial neural networks are the most powerful learning models in the field of machine learning inspired by the human brain.

In the past few years, deep artificial neural networks have proven to perform surprisingly well for complex tasks such as speech recognition (converting speech to text), machine translation,image and video classification. Such models are also commonly called deep learning models.

A biological neuron works as follows: it receives signals through its dendrites that are either amplified or inhibited as they pass through the axons to the dendrites of other neurons.

Artificial neural networks are a collection of many simple devices called artificial neurons. The network 'learns' to conduct certain tasks, such as recognising a cat, by training the neurons to 'fire' in a certain way when given a particular input, such as a cat. In other words, the network learns to inhibit or amplify the input signals to perform a certain task, such as recognising an animal, speaking a word, identifying a tree.

The applications of neural networks are across various domains such as images and videos (computer vision), text, speech. The terms 'deep learning' and 'neural networks' are often used interchangeably.

Perceptron

A perceptron acts like a tool that enables you to predict outcome based on multiple factors. Each decision factor holds a different 'weight'. Take different factors as input signals, attach a weight based on importance they attach to the corresponding factors and perform basic operations to get an output.

In other terms, the perceptron takes a weighted sum of multiple inputs (along with a bias) as the cumulative input and applies an output function on the cumulative input to get the output, which then assists in making a decision.

$$\text{Cumulative Input} = w_1x_1 + w_2x_2 + w_3x_3 + b$$

Where, w_i's represent the inputs, w_i's represent the weights associated with inputs and b is the bias.

$$w = \begin{bmatrix} w_1 \\ w_2 \\ . \\ . \\ w_k \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_k \end{bmatrix}$$

A neat and concise way to represent the weighted sum of w and x is using the dot product of the transpose of the weight vector w^T and the input vector x. Now, let's understand this concept of taking the dot product of the transpose of the weight vector and the input vector.

The transpose of w is $w^T = [w_1 w_2 .... w_k]$ - a row vector of size 1 x k. Taking the dot product of w^T with x:

$$w^T.x = \begin{bmatrix} w_1 & w_2 & . & . & w_k \end{bmatrix}. \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_k \end{bmatrix} = w_1x_1 + w_2x_2 + ... + w_kx_k$$

After adding bias to $w^T.x$, you will get the following equation:

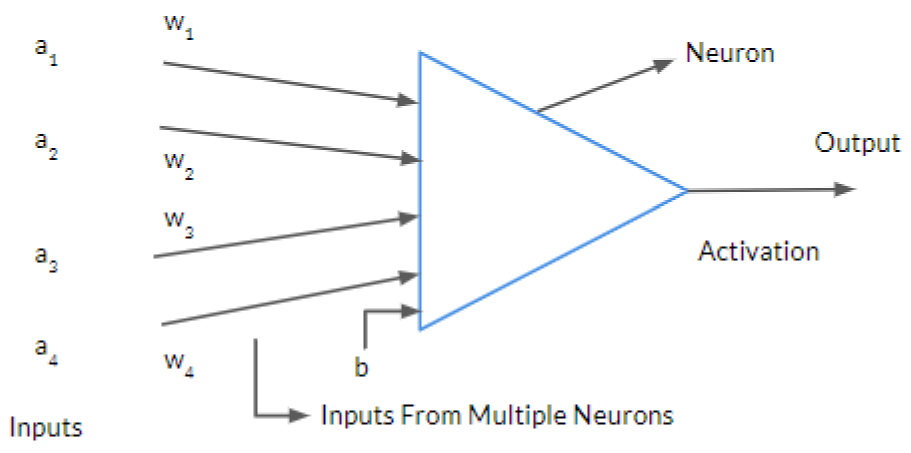
$$\text{Cumulative Input} = w^T.x + b = w_1x_1 + w_2x_2 + \ldots\ldots + w_kx_k + b$$

We then apply the step function to the cumulative input. According to the step function, if this cumulative weighted sum of inputs is > 0, the output is 1/yes; otherwise, it is 0/no.

Single Neuron

Neural networks are a collection of artificial neurons arranged in a particular structure. Now, you will learn how a single artificial neuron works. A neuron is very similar to a perceptron. In perceptrons, the activation function used is the step function, whereas in the case of ANNs, the activation functions are non-linear functions such as the sigmoid function.

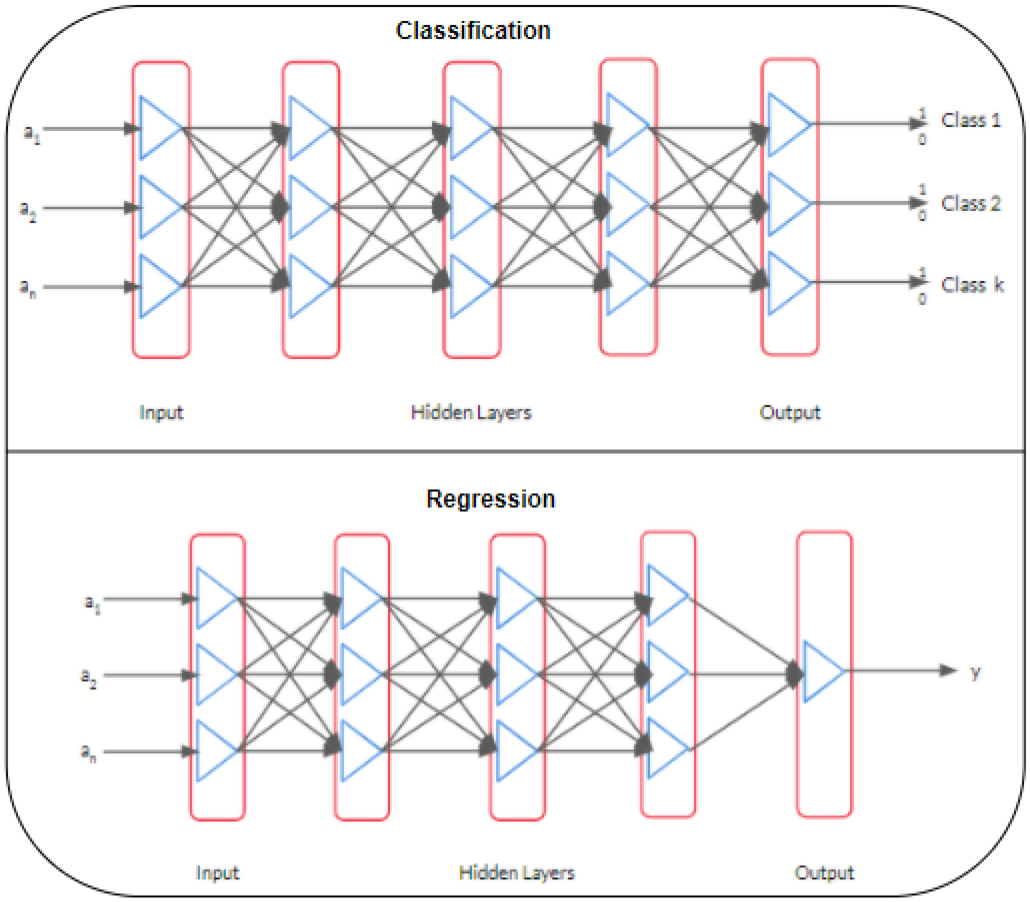
Please take a look at the structure of an artificial neuron in the image given below.



Where ‘a’ represents the inputs, ‘w’ represents the weights associated with the inputs and ‘b’ represents the bias of the neuron.

Multiple Artificial Neurons

In a neural network, multiple artificial neurons are arranged in different layers. The first layer is known as the input layer, and the last layer is called the output layer. The layers in between these two are the hidden layers. The number of neurons in the input layer is equal to the number of attributes/features in the data set, and those in the output layer are determined by the number of classes of the target variable (for a classification problem). For a regression problem, the number of neurons in the output layer is 1 (a numeric value). Please take a look at the image given below to understand the topology of neural networks in the case of classification and regression problems.



Basic Structure of Artificial Neural Networks

To summarise, the six main things that must be specified for any neural network are as follows:

1. Input layer
2. Output layer
3. Hidden layers
4. Network topology or structure
5. Weights and biases

6. Activation functions

Input

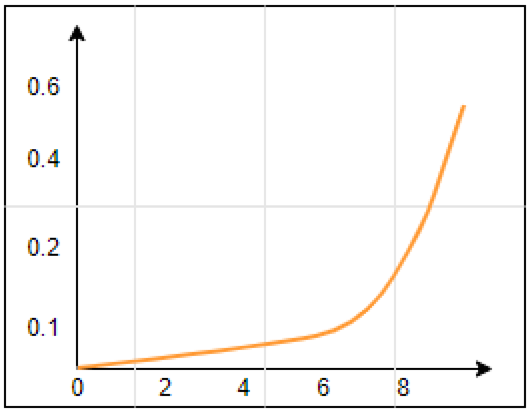
The most important point to notice is that the inputs can only be numeric. For different types of input data, you can use different ways to convert the inputs to a numeric form. The commonly used inputs used for ANNs are given below:

1. **Structured data:** The type of data that we use in standard machine learning algorithms have multiple features and are available in two dimensions such that the data can be represented in a tabular format. This type of data can be used as an input for training ANNs.
2. **Text data:** For text data, you can use a one-hot vector or word embeddings corresponding to a certain word.
3. **Image:** Images are naturally represented as arrays of numbers and can, thus, be fed into the network directly. These numbers are the raw pixels of an image. In images, pixels are arranged in rows and columns (an array of pixel elements).
4. **Speech:** In the case of a speech/voice input, the basic input unit is in the form of phonemes. These are distinct units of speech in any language. The speech signal is in the form of waves, and to convert these waves into numeric inputs, you need to use Fourier Transform. The input after this conversion will be numeric; so, you will be able to feed it into a neural network.

Output

Depending on the nature of the task, the outputs of neural networks can either be in the form of classes (if it is a classification problem) or numeric (if it is a regression problem).

One of the commonly used output functions is the softmax function for classification. Please take a look at the graphical representation of the softmax function shown below.



A softmax output is similar to what we get from a multiclass logistic function commonly used to compute the probability of an output belonging to one of the multiple classes. It is given by the following formula:

$$p_i = \frac{e^{w_i.x'}}{\sum_{t=0}^{c-1} e^{w_t.x'}}$$

Where c is the number of classes or neurons in the output layer, x' is the input to the network and wi's are the weights associated with the inputs.

Let's consider the case where the output layer has three neurons, and all of them have the same input x' (coming from the previous layers in the network). The weights associated with them are represented as w0, w1 and w2. In such a case, the probability of the input belonging to each of the classes is as follows:

$$p_0 = \frac{e^{w_0 x'}}{e^{w_0 x'} + e^{w_1 x'} + e^{w_2 x'}} \quad p_1 = \frac{e^{w_1 x'}}{e^{w_0 x'} + e^{w_1 x'} + e^{w_2 x'}} \quad p_2 = \frac{e^{w_2 x'}}{e^{w_0 x'} + e^{w_1 x'} + e^{w_2 x'}}$$

Also,from these expressions,it is evthat the sum of p0 + p1 + p2 is 1 and that p0, p1, p2 ϵ (0,1).

We have seen the softmax function as a commonly used output function in multiclass classification. Now, you will learn how this function translates to the sigmoid function in the special case of binary classification.

In the case of a sigmoid output, only one neuron is present in the output layer since if there are two classes with probabilities p0 and p1, we know that p0 + p1 = 1. Hence, we need to compute the value either of p0 or p1. In other words, the sigmoid function is just a special case of the softmax function (since binary classification is a special case of multiclass classification).

We can derive the sigmoid function from the softmax function as shown below. Let's assume that the softmax function has two neurons with the following outputs:

$$p_0 = \frac{e^{w_0 \cdot x'}}{e^{w_0 \cdot x'} + e^{w_1 \cdot x'}} \text{ and } p_1 = \frac{e^{w_1 \cdot x'}}{e^{w_0 \cdot x'} + e^{w_1 \cdot x'}}$$

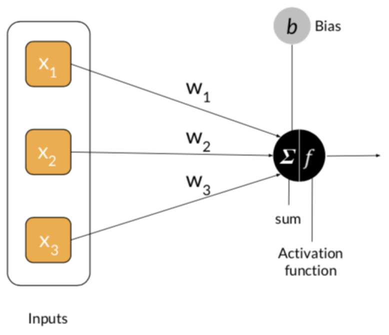
Consider only p1 and divide the numerator and the denominator with the numerator. We can now rewrite p1 as :

$$p_1 = \frac{1}{1 + \frac{e^{w_0 \cdot x'}}{e^{w_1 \cdot x'}}} = \frac{1}{1 + e^{(w_0 - w1) \cdot x'}}$$

If we assign (w1-w0) with new w , we get the sigmoid function.

Internal Workings of a Neuron

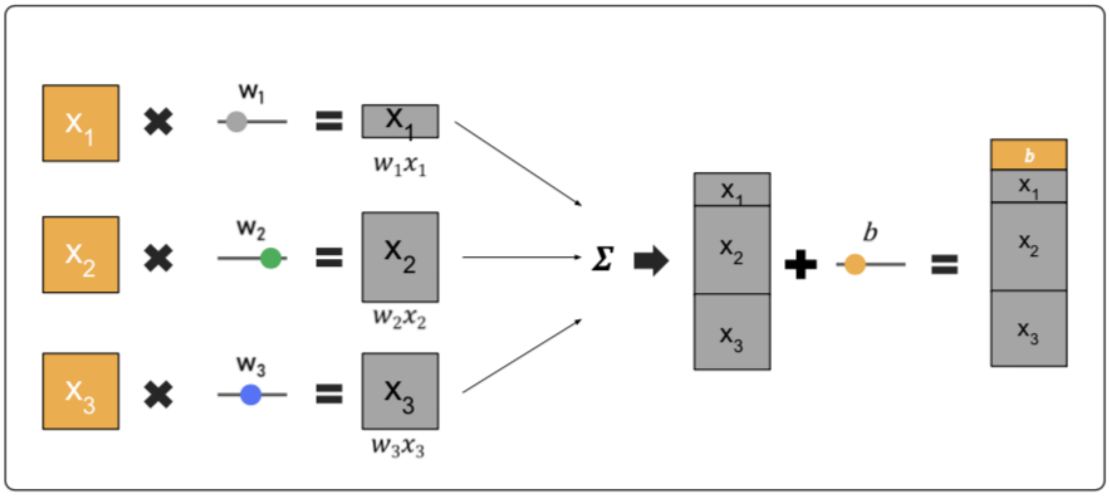
The visual representation of how inputs are fed into a neuron and how we obtain outputs using activation functions is shown below.



In the image, you can see that x1, x2 and x3 are the inputs, and their weighted sum along with a bias is fed into the neuron to give the calculated result as the output.

The weights are applied on each of the inputs, and along with the bias, the cumulative input is fed to the neuron. An activation function is then applied to the cumulative input to obtain the neuron's output. In the previous segment, you learnt about some of the activation functions such as softmax and sigmoid. In the next segment, we will explore more types of activation function. These functions apply non-linearity to the cumulative input to enable the neural network to identify complex non-linear patterns present in the data.

An in-depth representation of a cumulative input(Z) is shown below.



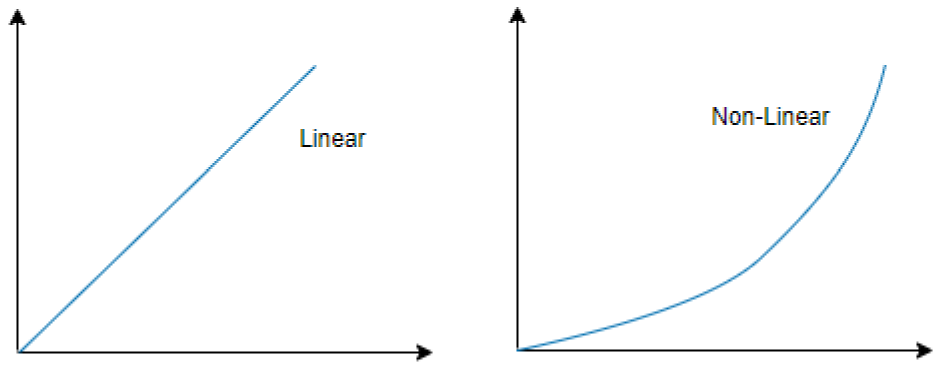
$$z = w_1x_1 + w_2x_2 + w_3x_3 + b$$

In this image, z is the cumulative input. You can see how the weights affect the inputs depending on their magnitudes. z is the dot product of the weights and inputs plus the bias.

Previously, you have learnt how a neuron takes an input and performs some operations on it to give the output. The output is obtained through an activation function.

Activation functions introduce non-linearity in the network, thus making the network capable of solving very complex problems. Problems for which the help of neural networks is to be taken require the ANN to recognise complex patterns and trends in data. If no non-linearity was introduced, it would result in the output being just a linear function of the input vector. This will not help us in understanding more complex patterns present in the data.

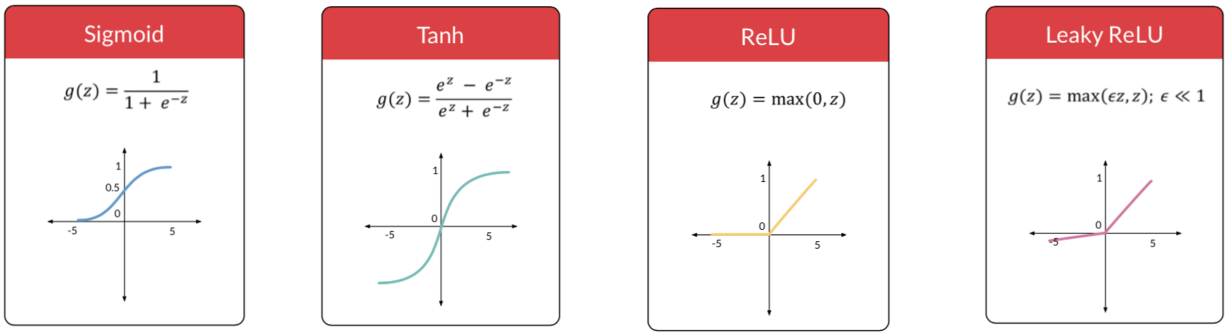
Please take a look at the image provided below that shows the graphical representation of a linear function and one of the possible representations of a non-linear function.



The main conditions that you need to keep in mind while choosing activation functions are that they should be:

- Non-linear
- Continuous
- Monotonically increasing

The different commonly used activation functions are given below.



Output values bound between **0 and 1.** Outputs not zero-centered.

Output values bound **between -1 and 1.** Outputs are **zero-centered**

Computationally efficient: allows the network to converge very quickly. **Dead ReLU:** when the value is negative, no learning happens

Allows learning even when the value is negative.

1. Sigmoid
2. Hyperbolic Tangent (Tanh)
3. Rectified Linear Unit (ReLU)
4. Leaky ReLU (Leaky Rectified Linear Unit (Leaky ReLU)

Parameters and Hyperparameters of Neural Networks

During training, the neural network learning algorithm fits various models to the training data and selects the best model. The learning algorithm is trained with a predefined fixed set of hyperparameters associated with a network structure. Some of these are given below:

- Number of layers
- Number of neurons in the input, hidden and output layers
- Learning rate (the step size taken each time we update the weights and biases of an ANN)
- Number of epochs (the number of times the entire training data set passes through the neural network)

The purpose of training is to obtain optimum weights and biases, which form the parameters of the network.

The notations that you will come across are as follows:

1. *W* represents the weight matrix.
2. *b* stands for bias.
3. *x* represents the input.
4. *y* represents the ground truth label.
5. *p* represents the probability vector of the predicted output for the classification problem. h^L represents the predicted output for the regression problem (where L represents the number of layers).
6. *h* also represents the output of the hidden layers with appropriate superscript. The output of the second neuron in the nth hidden layer is denoted by h_2^n.
7. z represents the accumulated input to a layer. The accumulated input to the third neuron of the nth hidden layer is z_3^n.
8. The bias of the first neuron of the third layer is represented as b_1^3.
9. The superscript represents the layer number. The weight matrix connecting the first hidden layer to the second hidden layer is denoted by W^2.
10. The subscript represents the index of an individual neuron in a given layer. The weight connecting the first neuron of the first hidden layer to the third neuron of the second hidden layer is denoted by w_{31}^2.

Assumptions for Simplifying Neural Networks

Commonly used neural network architectures make the following simplifying assumptions:

1. The neurons in an ANN are arranged in layers, and these layers are arranged sequentially.
2. The neurons within the same layer do not interact with each other.
3. Inputs are fed to the network through the input layer, and the outputs are sent out from the output layer.
4. Neurons in consecutive layers are densely connected, i.e., all neurons in layer l are connected to all neurons in layer l+1.
5. Every neuron in a neural network has a bias associated with it, and each interconnection has a weight associated with it.

6. All neurons in a particular hidden layer use the same activation function. Different hidden layers can use different activation functions, but in a hidden layer, all neurons use the same activation function.