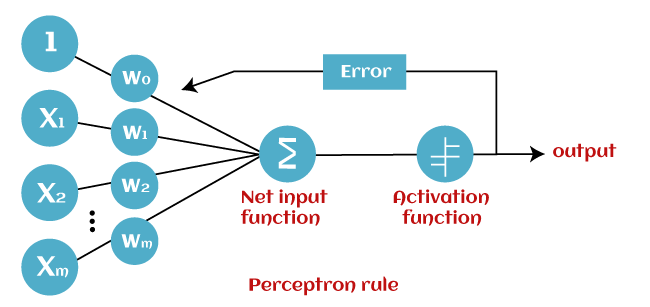
**AI&ML CIE-5 Key Answers**

**Section – I**

1. **a) Explain the working of Artificial neuron /Perceptron.**

In Machine Learning, Perceptron is considered as a single-layer neural network that consists of four main parameters named input values (Input nodes), weights and Bias, net sum, and an activation function. The perceptron model begins with the multiplication of all input values and their weights, then adds these values together to create the weighted sum. Then this weighted sum is applied to the activation function **'f'** to obtain the desired output. This activation function is also known as the **step function** and is represented by **'f'**.



This **step function or Activation function** plays a vital role in ensuring that output is mapped between required values (0,1) or (-1,1). It is important to note that the weight of input is indicative of the strength of a node. Similarly, an input's bias value gives the ability to shift the activation function curve up or down.

**b) Write a note on Activation function.**

**Activation function:**

* It’s a function that we use to get the output of node. It is also known as **Transfer Function**.
* It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function).
* The Activation Functions can be basically divided into 2 types:
  + Linear Activation Function
  + Non-linear Activation Function

**Linear Activation Function:** As we can see the function is a line or linear. Therefore, the output of the functions will not be confined between any range.

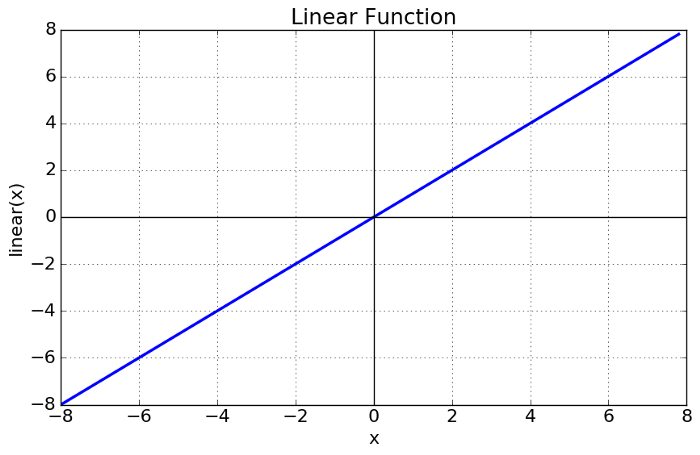


Fig: Linear Activation Function

* **Equation:** f(x) = x
* **Range:** (-infinity to infinity)
* It doesn’t help with the complexity or various parameters of usual data that is fed to the neural networks.

**Non-linear Activation Function:** The Nonlinear Activation Functions are the most used activation functions. Nonlinearity helps to makes the graph look something like this



Fig: Non-linear Activation Function

* It makes it easy for the model to generalize or adapt with variety of data and to differentiate between the output.
* The main terminologies needed to understand for nonlinear functions are:
  + **Derivative or Differential:** Change in y-axis w.r.t. change in x-axis. It is also known as slope.
  + **Monotonic function:** A function which is either entirely non-increasing or non-decreasing.

1. **Explain tokenization in NLP with example**.

Token in a text document refers to each “entity” that is a part of whatever was split up based on rules. For examples, each word is a token when a sentence is “tokenized” into words. Each sentence can also be a token, if you tokenized the sentences out of a paragraph. So basically, tokenizing involves splitting sentences and words from the body of the text.

1. **Sentence tokenization:** The given document is divided or tokenized into sentences.

**Syntax:** nltk.tokenize.sent\_tokenize(text, language='english')

returns a sentence-tokenized copy of text, using NLTK’s recommended sentence tokenizer. (currently PunktSentenceTokenizer for the specified language).

**Parameters:** Text (str) – text to split into sentences Language(str) – the model name in the Punkt corpus. By default it will be English.

**Example:**

from nltk.tokenize import sent\_tokenize

EXAMPLE\_TEXT = "Hello Mr. Smith, how are you doing today? The weather is great, and Python is awesome. The sky is pinkish-blue. You shouldn't eat cardboard."

print(sent\_tokenize(EXAMPLE\_TEXT))

1. **Word tokenization**: The given document is divided or tokenized into words. **Syntax:** nltk.tokenize.word\_tokenize(text, language='english', preserve\_line=False)

returns a tokenized copy of text, using NLTK’s recommended word tokenizer (currently an improved TreebankWordTokenizer along with PunktSentenceTokenizer for the specified language).

**Parameters text (str)** – text to split into words language (str) – the model name in the Punkt corpus preserve\_line (bool) – A flag to decide whether to sentence tokenize the text or not.

**Example:**

from nltk.tokenize import word\_tokenize

EXAMPLE\_TEXT = "Hello Mr. Smith, how are you doing today? The weather is great, and Python is awesome. The sky is pinkish-blue. You shouldn't eat cardboard."

print(word\_tokenize(EXAMPLE\_TEXT))

**Section-2**

1. **Build a Neural Network in python for MNIST dataset to classify handwritten numerals. Create a git repository and push source code to repo.**

**Code:**

import tensorflow as tf

from tensorflow import keras

import matplotlib.pyplot as plt

%matplotlib inline

import numpy as np

(X\_train, y\_train) , (X\_test, y\_test) = keras.datasets.mnist.load\_data()

len(X\_train)

len(X\_test)

X\_train[0].shape

y\_train[0]

X\_train = X\_train / 255

X\_test = X\_test / 255

X\_train\_flattened = X\_train.reshape(len(X\_train), 28\*28)

X\_test\_flattened = X\_test.reshape(len(X\_test), 28\*28)

X\_train\_flattened.shape

model = keras.Sequential([

keras.layers.Flatten(input\_shape=(28, 28)),

keras.layers.Dense(100, activation='relu'),

keras.layers.Dense(10, activation='sigmoid')

])

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

tb\_callback = tf.keras.callbacks.TensorBoard(log\_dir="logs/", histogram\_freq=1)

model.fit(X\_train, y\_train, epochs=5, callbacks=[tb\_callback])

%load\_ext tensorboard

%tensorboard --logdir logs/