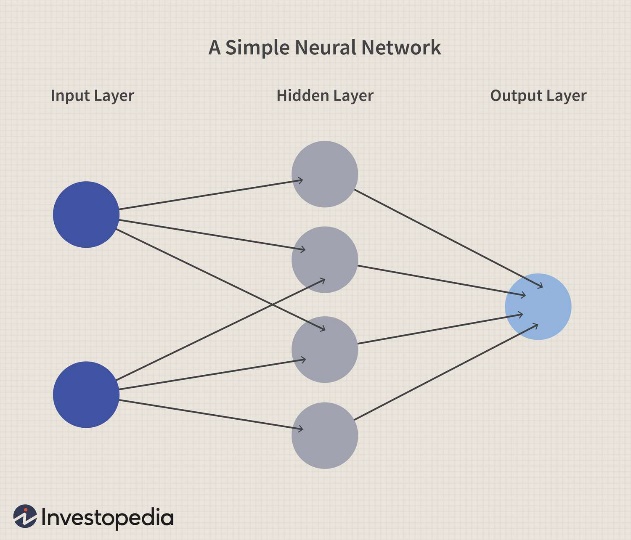
Deep Learning

# Introduction to Neural networks & Deep Learning

* Basic Neural Network architecture:



* First layer of neural networks is input layer
* input layer will have nodes which will have features
* hidden layer can have any number of neurons
* As information goes through all the neurons, preprocessing happens in the hidden layer, and we get the output.
* The above resembles artificial neural networks (ANN)
* To understand better, consider an example of distinguishing a cat & a dog.
* Output can have multiple nodes as well.

# How does neural network work?

* Suppose that we have x1, x2, x3 in input layer & we are trying to solve a binary classification problem
* + bias
* )
* Hidden layer neurons does some processing , some weights are assigned before passing through the neurons (there weights & activation functions will be playing vey important roles.
* two operations take place in hidden neuron:
* summation of wights and input features.
* activation function applied

Act ()

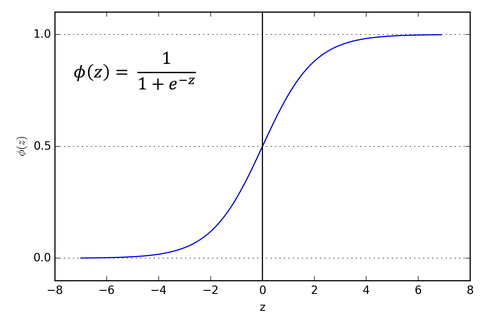
* we multiply weights and input features and sum them and then apply activation function
* there are many activation functions like sigmoid activation function ( )
* activation function will transform the values between 0 & 1.
* Neurons will get activated for higher weights
* Again, from going/passing through neuron to output layer there is weight assigned to it (w4)
* Z= z\*w4
* = Act(z)
* So w4 will get multiplied by z & activation function will be applied & we get binary output.
* This is an example of forward propagation.
* We can have multiple neurons and all neurons will do the same thing.

# Activation functions :-

* 1) sigmoid activation function
* 2)Relu activation function
* As discussed in each feature will get multiplied to the weights

(bi= bias)

* Activation functions transform y into 0 to 1.
* Sigmoid activation function:

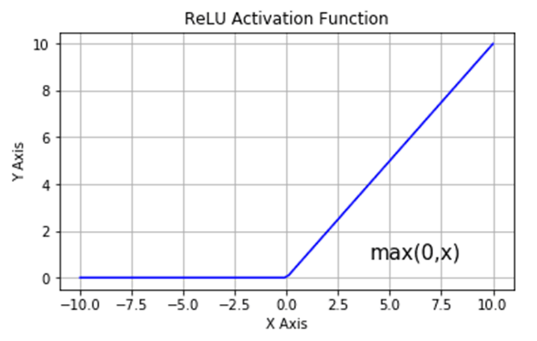


* This function takes any real value as input & outputs values in range 0 & 1, larger the input (more positive) the closer the output value will be 1.0 , whereas the smaller the input (more negative), the closer the output to 0.0.
* We calculate z as summation of weights and input features and apply activation function on them.
* If value < 0.5 ----------- 0 (neurons not activated)

If value > 0.5 ----------- 1 (neurons deactivated)

So activated neurons transfer he signal & helps in classification.

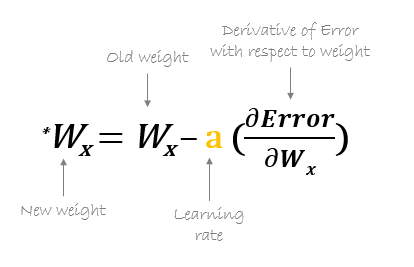
* ReLU activation function:



* Relu activation function is applied then we apply a simple formula (max(z,0))
* If y is negative, it gets transformed to 0 & if it greater than 0 then we take that value.
* This is more popular activation function.

# How to train neural network with backpropagation?

* In general, in a neural network all the features will get passed to hidden neurons in the hidden layer, all the features will get multiplied by weights + bias & then activation function is applied. Once the features are getting transformed it will go and pass to output layer, where weights are assigned. Yhat will be the predicted output and we compare y & yhat
* Weights must be adjusted in such a way that yhat should match y and loss value must be reduced.
* This can be done by using an optimizer like gradient descent & stochastic gradient descent.
* To reduce the loss value, we must backpropagate and must update the weights.

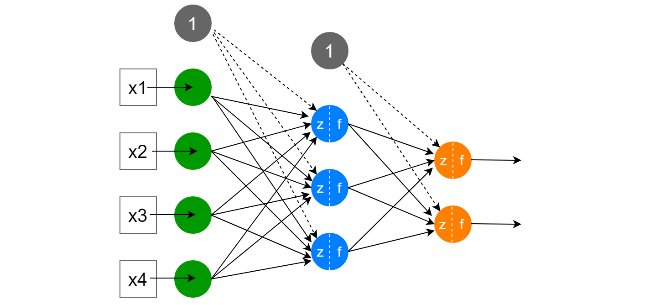


* (Derivative instantaneous rate of change of a function with respect to another variable.)
* Once we update the weight w4, we update w1,w2,w3 as well.
* After updating all the weights front propagation will go ahead and same process will be followed, once the loss value reduced, y & yhat will match.
* This is about training neural networks.
* For single record we have derived the loss function, but if we have multiple records at that time, we must define a cost function.

Cost function =

* Once we get the loss function value the main aim will be to reduce the loss function value with the optimizer.
* Optimizer will help us to find out derivative & it will try to change the values.
* Learning rate should be not too small & not too large.

# How to train multilayer neural networks & gradient descent



(Consider only one output layer for example understanding as below)

* First need to understand what kind of matrix gets formed & the hidden layer with the input.
* Weights will get multiplied with input features & all summation will happen & bias gets added , then activation function will be applied. Similar will happen in the next layer.
* We get output yhat.
* yhat = predicted value, y= actual value

# 27. Create CNN model and optimize it using Keras:

!pip install keras-tuner

import tensorflow as tf

from tensorflow import keras

import numpy as np

print(tf.\_\_version\_\_)

# we will be using fashion mnist dataset

#it has various fashion related dataset

#we will import it from tensorflow

fashion\_mnist = keras.datasets.fashion\_mnist

#create train and test images

(train\_images,train\_labels),(test\_images,test\_labels) = fashion\_mnist.load\_data()

#we scale the data, as data is in grey scale, we scale it for training purposes

train\_images = train\_images/255.0 #maximum pixels that can be present

test\_images = test\_images/255.0

#for CNN we need to reshape the data

#check the first image

train\_images[0]

train\_images[0].shape #here we have 28 by 28 pixels

#we need to resize it

train\_images = train\_images.reshape(len(train\_images),28,28,1)

#we do the same thing for test data, reshape it

test\_images = test\_images.reshape(len(test\_images),28,28,1)

def build\_model(hp):  #hp = hyperparameters

  model = keras.Sequential([

      keras.layers.Conv2D(

          filters = hp.Int('conv\_1\_filter', min\_values = 32, max\_values = 128, step = 16), #create integers , minimum value of filters = 32, max = 128

          kernel\_size = hp.choice('conv\_1\_kernel', values = [3,5]), #filter size

          activation = 'relu',

          input\_shape = (28,28,1)

      ),  #for each and every values we will check , it will help us to find out which convolution 2D layer can use the exact filter size

       keras.layers.Conv2D(

          filters = hp.Int('conv\_2\_filter', min\_values = 32, max\_values = 64, step = 16), #2nd layer

          kernel\_size = hp.choice('conv\_2\_kernel', values = [3,5]), #filter size

          activation = 'relu'

      ),

      keras.layers.Flatten(),  #we can flatten tha layer

      keras.layers.Dense(

          units = hp.Int('dense\_1\_units', min\_values = 32, max\_values = 128, step = 16),

          activation = 'relu'

      ),

      keras.layers.Dense(10,activation = 'softmax')    #number of nodes to select in Dense

  ])

  #choice also helps us to select range of parameters

  model.compile(optimizer = keras.optimizers.Adam(hp.Choice('learning\_rate',values = [1e-2,1e-3])),  #we have to provide learning rate for adam

                loss = 'sparse\_categorical\_crossentropy',

                metrics = ['accuracy'])

  return model

from kerastuner import RandomSearch

from kerastuner.engine.hyperparameters import hyperparameters