NAME: AYUSH CHANCHAL

SAP ID: 500097569

Batch: 06

EXPERIMENT:02

1. Explain the importance of Activation Function in Neural Network.

: - An activation function decides the input of a neuron to the network is important or not in the process of prediction while using mathematical operations. It is use to convert neurons in non-linear to predict the pattern and on the basis of that model will predict the output.

Types of Activation Function:

1. Sigmoid Function: In this type of activation function, we give the input any real number and we get the output in the range of 0 to 1.

Formula:

f(x)=1/(1+exp-x)

x=W0x1+w1x2+w2x3……..

1. Tanh Function (Hyperbolic Tangent): This Function is very similar to sigmoid function because it has the same S graph as the sigmoid function but it differs in the range. It take the real number as input but the output range is from -1 to +1.

Formula:

f(x) = (ex – e-x) / (ex + e-x)

x=W0x1+w1x2+w2x3……..

1. RELU Function (Rectified Linear Unit): It has a derivative function and allows for backpropagation while simultaneously making it computationally efficient. If the output is less than 0 then only the neurons will be deactivated.

Formula:

f(x): max (0, x)

x=W0x1+w1x2+w2x3……..

1. Leaky RELU Function: It is the modified version of RELU Function. Instead of defining the RELU activation function as 0 for negative values of inputs(x), we define it as an extremely small linear component of x.

Formula:

f(x)= max (0.01\*x, x)

x=W0x1+w1x2+w2x3……..

1. SoftMax Function: It is the last Activation function of neural network and it is use to normalize the output of a network to a probability distribution over predicted output class.

Formula:

S(y)i= exp(yi)/(∑j=1 exp(yj))

Where,

y= input value

yi= i-th element of the input vector

∑exp(yi)= a normalization value

1. Implementing the popular activation functions.

I). Sigmoid Function:

Source code:

import matplotlib.pyplot as plt

import numpy as np

def sigmoid(x):

s=1/(1+np.exp(-x))

ds=s\*(1-s)

return s,ds

x=np.arange(-6,6,0.01)

sigmoid(x)

fig, ax = plt.subplots(figsize=(9, 5))

ax.spines['left'].set\_position('center')

ax.spines['right'].set\_color('none')

ax.spines['top'].set\_color('none')

ax.xaxis.set\_ticks\_position('bottom')

ax.yaxis.set\_ticks\_position('left')

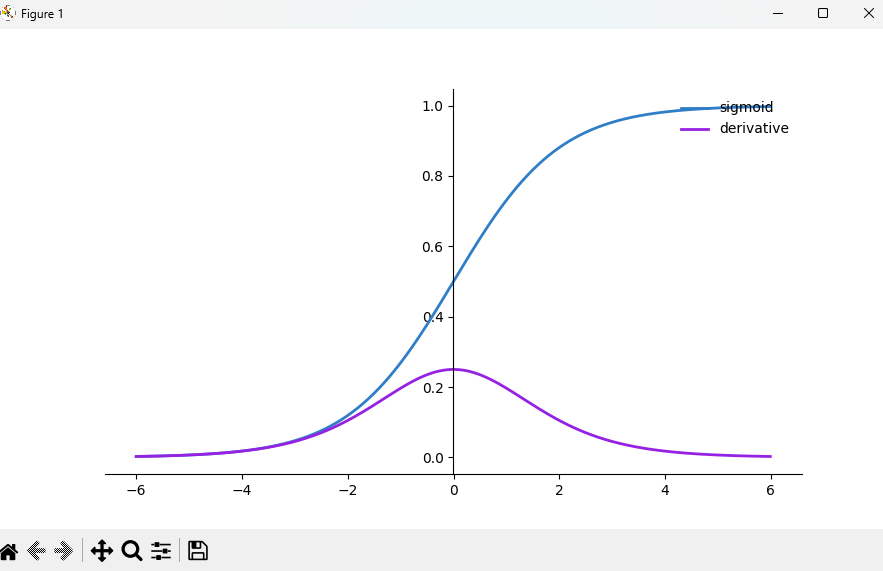
ax.plot(x,sigmoid(x)[0], color="#307EC7", linewidth=2, label="sigmoid")

ax.plot(x,sigmoid(x)[1], color="#9621E2", linewidth=2, label="derivative")

ax.legend(loc="upper right", frameon=False)

plt.show()

Graph:



II). Tangent Hyperbolic Function

Source code:

import matplotlib.pyplot as plt

import numpy as np

def tanh(x):

t=(np.exp(x)-np.exp(-x))/(np.exp(x)+np.exp(-x))

dt=1-t\*\*2

return t,dt

z=np.arange(-6,6,0.01)

tanh(z)[0].size,tanh(z)[1].size

fig, ax = plt.subplots(figsize=(9, 5))

ax.spines['left'].set\_position('center')

ax.spines['bottom'].set\_position('center')

ax.spines['right'].set\_color('none')

ax.spines['top'].set\_color('none')

ax.xaxis.set\_ticks\_position('bottom')

ax.yaxis.set\_ticks\_position('left')

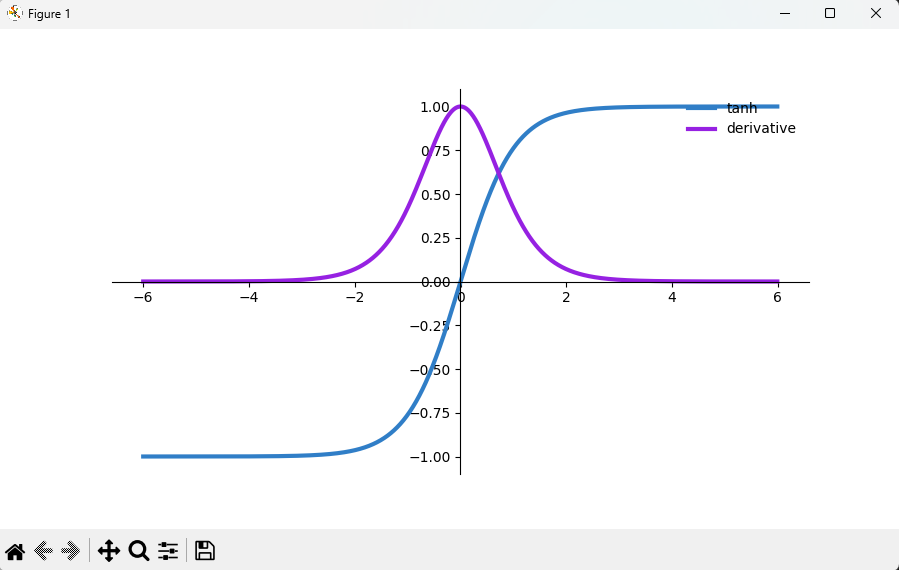
ax.plot(z,tanh(z)[0], color="#307EC7", linewidth=3, label="tanh")

ax.plot(z,tanh(z)[1], color="#9621E2", linewidth=3, label="derivative")

ax.legend(loc="upper right", frameon=False)

plt.show()

Graph:



III). RELU Function

Source code:

import numpy as np

import matplotlib.pyplot as plt

def ReLU(x):

data = [max(0,value) for value in x]

return np.array(data, dtype=float)

def der\_ReLU(x):

data = [1 if value>0 else 0 for value in x]

return np.array(data, dtype=float)

x\_data = np.linspace(-10,10,100)

y\_data = ReLU(x\_data)

dy\_data = der\_ReLU(x\_data)

plt.plot(x\_data, y\_data, x\_data, dy\_data)

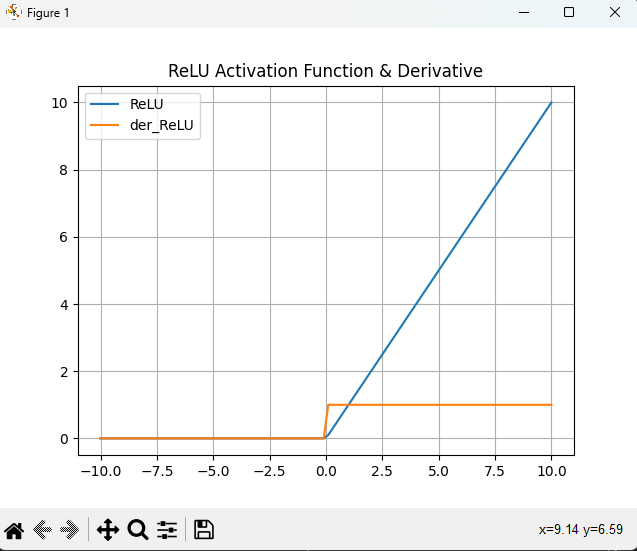
plt.title('ReLU Activation Function & Derivative')

plt.legend(['ReLU','der\_ReLU'])

plt.grid()

plt.show()

Graph:



IV). Leaky RELU Function

Source code:

import numpy as np

import matplotlib.pyplot as plt

def leaky\_ReLU(x):

data = [max(0.05\*value, value) for value in x]

return np.array(data, dtype=float)

def der\_leaky\_ReLU(x):

data = [1 if value > 0 else 0.05 for value in x]

return np.array(data, dtype=float)

x\_data = np.linspace(-10, 10, 100)

y\_data = leaky\_ReLU(x\_data)

dy\_data = der\_leaky\_ReLU(x\_data)

plt.plot(x\_data, y\_data, x\_data, dy\_data)

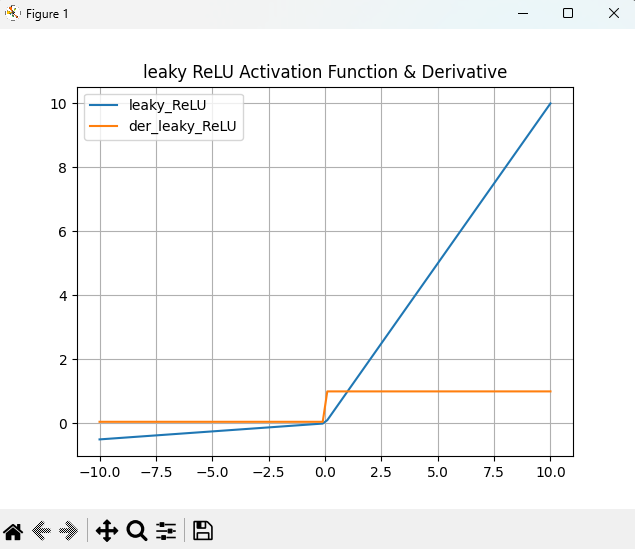
plt.title('leaky ReLU Activation Function & Derivative')

plt.legend(['leaky\_ReLU', 'der\_leaky\_ReLU'])

plt.grid()

plt.show()

Graph:



V). SoftMax Function

Source code:

import matplotlib.pyplot as plt

import numpy as np

def softmax(x):

''' Compute softmax values for each sets of scores in x. '''

return np.exp(x) / np.sum(np.exp(x), axis=0)

x = np.linspace(-10, 10)

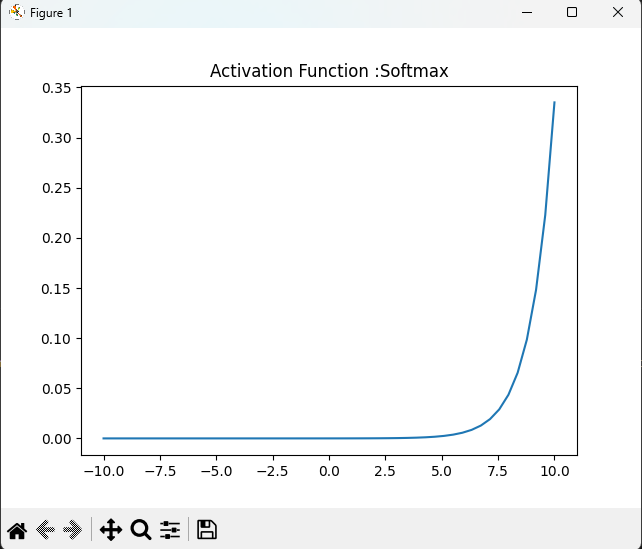
plt.plot(x, softmax(x))

plt.axis('tight')

plt.title('Activation Function :Softmax')

plt.show()

Graph:



1. Advantages and Disadvantages of the types of Activation Function.
2. Sigmoid function

Advantages:

* Output values range between 0 and 1, mimicking a probability distribution.
* It is differentiable, making it suitable for gradient-based optimization.

Disadvantages:

* Prone to the vanishing gradient problem, which can slow down or halt learning in deep networks.
* Outputs are not zero-centered, which may hinder learning

1. Hyperbolic Tangent (tanh) Activation Function:

Advantages:

* Similar to the sigmoid function, but zero-centered, which can speed up convergence.
* It is differentiable.

Disadvantages:

* Still susceptible to the vanishing gradient problem.
* The output is bounded between -1 and 1, which may not be appropriate for all scenarios.

1. Rectified Linear Unit (ReLU):

Advantages:

* Computationally efficient due to its simple thresholding.
* Helps mitigate the vanishing gradient problem.
* Promotes sparse activations, making the network easier to optimize.

Disadvantages:

* Can suffer from the "dying ReLU" problem, where neurons get stuck and cease to learn.

1. Leaky ReLU:

Advantages:

* Addresses the "dying ReLU" problem by allowing a small gradient for negative inputs.
* Helps with the vanishing gradient problem.

Disadvantages:

Adds complexity with an extra hyperparameter (slope of the negative part).