import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

read\_data\_file = pd.read\_csv('/content/A2\_Q4\_data.csv')

#convert .csv file data to numpy array

read\_data\_file=np.array((read\_data\_file))

# seperate two columns

# input col\_1 = x

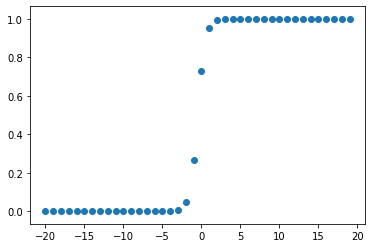
# input col\_2 = y

col\_1=read\_data\_file[:,0]

col\_2=read\_data\_file[:,1]

#plot given data

plt.scatter(col\_1,col\_2)



#initialize variable

learning\_parameter= 0.01

w=1

b=1

total\_loss = 0

X = col\_1

Y = col\_2

# create sigmoid function

def sigmoid\_val(w,b,x):

    return 1.0/(1.0 + np.exp(-(w\*x+b)))

# create loss function

def error (w,b):

    err=0.0

    for x,y in zip(X,Y):

        fx = sigmoid\_val(w,b,x)

        err += 0.5 \* (fx - y) \*\* 2

    return err

def grad\_b(w,b,x,y):

    fx = sigmoid\_val(w,b,x)

    return (fx - y) \* fx \* (1 - fx)

def grad\_w(w,b,x,y):

    fx = sigmoid\_val(w,b,x)

    return (fx - y) \* fx \* (1 - fx) \* x

err\_val = []

for i in range(100):

    dw = 0

    db = 0

    count=error(w,b)

    err\_val.append(count)

    for x,y in zip(X,Y):

        dw += grad\_w(w,b,x,y)

        db += grad\_b(w,b,x,y)

    # find value of w and b

    w = w - learning\_parameter \* dw

    b = b - learning\_parameter \* db

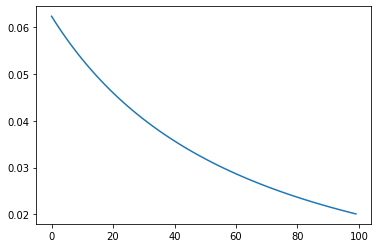
print('value of w : ',w)

print('value of b : ',b)

value of w : 1.1527532551317377

value of b : 0.9194429387731992

plt.plot(err\_val)



Another gradient descent

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

read\_data\_file = pd.read\_csv('/content/A2\_Q4\_data.csv')

read\_data\_file=np.array((read\_data\_file))

col\_1=read\_data\_file[:,0]

col\_2=read\_data\_file[:,1]

plt.scatter(col\_1,col\_2)

eta= 0.1

w=1

b=1

total\_loss = 0

X = col\_1

Y = col\_2

def sigmoid\_val(w,b,x):

    return 1.0/(1.0 + np.exp(-(w\*x+b)))

def error (w,b):

    err=0.0

    for x,y in zip(X,Y):

        fx = sigmoid\_val(w,b,x)

        err += 0.5 \* (fx - y) \*\* 2

    return err

def grad\_b(w,b,x,y):

    fx = sigmoid\_val(w,b,x)

    return (fx - y) \* fx \* (1 - fx)

def grad\_w(w,b,x,y):

    fx = sigmoid\_val(w,b,x)

    return (fx - y) \* fx \* (1 - fx) \* x

err\_val = []

update\_wplt=[]

update\_bplt=[]

wplt=[]

bplt=[]

prevupdate\_w,prevupdate\_b,gamma=0,0,0.9

for i in range(100):

    dw = 0

    db = 0

    print('\nEpoch:\t',i+1)

    count=error(w,b)

    err\_val.append(count)

    for x,y in zip(X,Y):

        dw += grad\_w(w,b,x,y)

        db += grad\_b(w,b,x,y)

    update\_w=gamma\* prevupdate\_w + eta\*dw

    update\_b=gamma\* prevupdate\_b + eta\*db

    w = w -  update\_w

    b = b -  update\_b

    print('value of w : ',w)

    print('value of b : ',b)

    prevupdate\_w=update\_w

    prevupdate\_b=update\_b

    update\_wplt.append(update\_w)

    update\_bplt.append(update\_b)

    wplt.append(w)

    bplt.append(b)

print('Loss function:\t',err\_val)

plt.figure()

plt.plot(update\_wplt)

plt.figure()

plt.plot(update\_bplt)

plt.figure()

plt.plot(wplt)

plt.figure()

plt.plot(bplt)

Epoch: 1

value of w : 1.020294034406701

value of b : 0.9894130187778657

Epoch: 2

value of w : 1.0574403280898648

value of b : 0.9699740364944479

Epoch: 3

value of w : 1.1074077925415782

value of b : 0.9437341887772811

Epoch: 4

value of w : 1.1661845756717677

value of b : 0.9128076386175733

Epoch: 5

value of w : 1.2302008718033037

value of b : 0.8791742204678984

Epoch: 100

value of w : 2.0066123207920947

value of b : 0.9996238647591136