In [1]:
```
import numpy as np

print(np.__version__)
```

```
2.3.5
```

In [2]:
```
# Sigmoid activation function and its derivative

def sigmoid(x: np.ndarray) -> np.ndarray:
    return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x: np.ndarray) -> np.ndarray:
    sig = sigmoid(x)
    return sig * (1 - sig)
```

In [3]:
```
def RSS(y_true: np.ndarray, y_pred: np.ndarray) -> float:
    return np.sum((y_true - y_pred) ** 2)

def RSS_derivative(y_true: np.ndarray, y_pred: np.ndarray) -> np.ndarray:
    return 2 * (y_pred - y_true)
```

In [4]:
```
# Input dataset
X = np.array([[0, 0, 1],
              [0, 1, 1],
              [1, 0, 1],
              [1, 1, 1]])

# Output dataset
Y = np.array([[0],
              [0],
              [1],
              [1]])


# Set random seed for reproducibility
np.random.seed(1)
weights = np.random.normal(loc=0.0, scale=1/3, size=(3, 1))


# learning rate
lr = 1

# Declare RSS_res for print outside the loop scope
RSS_res = 0
```

In [5]:
```
# Training loop
for epoc in range(1000):
    # prod = sigmpoid(Input * Weights)
    input = X # 4x3
    pre_activation = np.dot(input, weights) # 4x1
    prod = sigmoid(pre_activation) # 4x1

    # error calculation
    RSS_res = RSS(Y, prod) # 4x1

    # The Chain role - (x1w1 + x2w2 +x3w3)` for each weight i 3x4  @  2 * (pred
    gradients = np.dot(input.T, RSS_derivative(Y, prod) * sigmoid_derivative(pre
    # Step - Update weights
    weights -= lr * gradients

    if epoc % 100 == 0:
        print(f"\n=========\nError after {epoc} iterations\nRSS = {RSS_res}\ngra
=========
Error after 0 iterations
RSS = 0.7521293108590142
gradients = [[-0.42650808]
 [-0.03245847]
 [-0.00430723]]
update weights = [[ 0.96795653]
 [-0.17146033]
 [-0.17175002]]
=========

=========
Error after 100 iterations
RSS = 0.015561273247217964
gradients = [[-0.01156335]
 [-0.00024253]
 [ 0.00598262]]
update weights = [[ 5.48381227]
 [-0.24916842]
 [-2.50228222]]
=========

=========
Error after 200 iterations
RSS = 0.0073340714502371195
gradients = [[-5.57945671e-03]
 [-8.67268126e-05]
 [ 2.85914188e-03]]
update weights = [[ 6.26792024]
 [-0.23489814]
 [-2.9059405 ]]
=========

=========
Error after 300 iterations
RSS = 0.004754547880877242
gradients = [[-3.65324488e-03]
 [-4.66841044e-05]
 [ 1.86358470e-03]]
update weights = [[ 6.71527344]
 [-0.22858226]
 [-3.13464353]]
=========

=========
Error after 400 iterations
RSS = 0.003505489890013448
gradients = [[-2.70914365e-03]
 [-2.99708236e-05]
 [ 1.37817875e-03]]
update weights = [[ 7.02828682]
 [-0.2248585 ]
 [-3.2940906 ]]
=========

=========
Error after 500 iterations
RSS = 0.0027714440508271785
gradients = [[-2.15019036e-03]
 [-2.12266639e-05]
 [ 1.09176511e-03]]
update weights = [[ 7.26884784]
 [-0.22234411]
 [-3.41634856]]
=========

=========
Error after 600 iterations
RSS = 0.002289309885631873
gradients = [[-1.78117327e-03]
 [-1.60055224e-05]
 [ 9.03132615e-04]]
update weights = [[ 7.46408584]
 [-0.22050524]
 [-3.51541028]]
=========

=========
Error after 700 iterations
RSS = 0.0019488202211708273
gradients = [[-1.51957136e-03]
 [-1.26043501e-05]
 [ 7.69653073e-04]]
update weights = [[ 7.6283063 ]
 [-0.21908755]
 [-3.59863104]]
=========

=========
Error after 800 iterations
RSS = 0.0016957543475006089
gradients = [[-1.32455663e-03]
 [-1.02474821e-05]
 [ 6.70293408e-04]]
update weights = [[ 7.76997265]
 [-0.21795282]
 [-3.67035214]]
=========

=========
Error after 900 iterations
RSS = 0.0015003832820970238
gradients = [[-1.17363761e-03]
 [-8.53707079e-06]
 [ 5.93491728e-04]]
update weights = [[ 7.89450507]
 [-0.21701872]
 [-3.73334871]]
=========
```

In [6]:

```python
print("\nFinal weights after training:")
print(weights)
print("final error:")
print(RSS_res)
```

```
Final weights after training:
[[ 8.00452918]
 [-0.21624012]
 [-3.7889692 ]]
final error:
0.0013464554282270162
```