In [1]:

```
import numpy as np
```

In [2]:

```
def init_network():
    layer_count=int(input('Enter number of hidden layers in Neural Network: '))+1
    layer_dict={}
    for i in range(layer_count+1):
        nodes = int(input(f'Enter of nodes in layer {i}: '))
        layer_dict['l'+str(i)]=nodes
    return layer_count, layer_dict
```

In [3]:

```
def init_data(layer_dict, layer_count):
    m=int(input('Enter number of input instances: '))
    X,Y=[[]*m],[[]*m]
    for i in range(m):
        for j in range(layer_dict['l0']):
            x_val = int(input(f'Enter feature value x{j+1} for instance {i}: '))
            X[i].append(x_val)
        y_val = int(input(f'Enter target label Y for instance {i}: '))
        Y[i].append(y_val)
    X=np.array(X).reshape([layer_dict['l0'],m])
    Y=np.array(Y).reshape([layer_dict['l'+str(layer_count)],m])
    return X, Y
```

In [4]:

```
def init_params(layer_dict,layer_count):
    parameters={}
    for i in range(layer_count):
        W=np.ones([layer_dict['l'+str(i+1)],layer_dict['l'+str(i)]])
        b=np.zeros([layer_dict['l'+str(i+1)],1])
        parameters['W'+str(i+1)]=W
        parameters['b'+str(i+1)]=b
    return parameters
```

In [5]:

```
def transform(weight,bias,x):
    return (np.dot(weight,x)+bias)
```

In [6]:

```
def forward_prop(X,parameters,layer_count):
    cache={}
    cache['A0']=X
    for i in range(layer_count):
        Z=transform(parameters['W'+str(i+1)],parameters['b'+str(i+1)], cache['A'+str(i)])
        A=Z
        cache['Z'+str(i+1)]=Z
        cache['A'+str(i+1)]=A
    return cache['A2'],cache
```

In [7]:

```
def calc_cost(final_output,target_labels):
    cost = np.square(np.subtract(target_labels,final_output))
    cost = np.mean(cost)
    return cost
```

In [8]:

```
def backward_prop(cache,parameters,X,Y,layer_count):
    m=cache['A0'].shape[1]
    updates={}
    dZ=1
    for i in reversed(range(1,1+layer_count)):
        if i==layer_count:
            dA=cache['A'+str(i)]-Y
        else:
            dA=dZ*np.multiply(dA.T,parameters['W'+str(i+1)])
            dW=(dZ*np.multiply(dA,cache['A'+str(i-1)]))/m
            db=(np.sum(dA*dZ))/m
            updates['dW'+str(i)]=dW
            updates['db'+str(i)]=db
    return updates
```

In [9]:

```
def gradient_descent(updates,parameters,learning_rate,layer_count):
    new_parameters={}
    for i in range(layer_count):
        new_parameters['W'+str(i+1)]=parameters['W'+str(i+1)]-learning_rate
        new_parameters['b'+str(i+1)]=parameters['b'+str(i+1)]-learning_rate
    return new_parameters
```

In [10]:

```
def model():
    layer_count, layer_dict = init_network()
    X, Y = init_data(layer_dict, layer_count)
    parameters = init_params(layer_dict,layer_count)
    learning_rate = float(input('Enter learning rate: '))
    num_iter = int(input('Enter number of iterations: '))
    for i in range(num_iter):
        output,cache = forward_prop(X, parameters, layer_count)
        cost = calc_cost(output, Y)
        print(f"\nCost after iteration {i+1}:",cost)
        updates = backward_prop(cache,parameters,X,Y,layer_count)
        parameters = gradient_descent(updates,parameters,learning_rate,layer_count)
        print(f"\nParameters after iteration {i+1}:",parameters)
```

In [11]:

```
model()
```

```
Enter number of hidden layers in Neural Network: 1
Enter of nodes in layer 0: 2
Enter of nodes in layer 1: 2
Enter of nodes in layer 2: 1
Enter number of input instances: 1
Enter feature value x1 for instance 0: 1
Enter feature value x2 for instance 0: 0
Enter target label Y for instance 0: 1
Enter learning rate: 0.01
Enter number of iterations: 10

Cost after iteration 1: 1.0

Parameters after iteration 1: {'W1': array([[0.99, 0.99],
       [0.99, 0.99]]), 'b1': array([[-0.01],
       [-0.01]]), 'W2': array([[0.99, 0.99]]), 'b2': array([[-0.01]])}

Cost after iteration 2: 0.8656441599999998

Parameters after iteration 2: {'W1': array([[0.98, 0.98],
       [0.98, 0.98]]), 'b1': array([[-0.02],
       [-0.02]]), 'W2': array([[0.98, 0.98]]), 'b2': array([[-0.02]])}

Cost after iteration 3: 0.7423545599999999

Parameters after iteration 3: {'W1': array([[0.97, 0.97],
       [0.97, 0.97]]), 'b1': array([[-0.03],
       [-0.03]]), 'W2': array([[0.97, 0.97]]), 'b2': array([[-0.03]])}

Cost after iteration 4: 0.6298009599999997

Parameters after iteration 4: {'W1': array([[0.96, 0.96],
       [0.96, 0.96]]), 'b1': array([[-0.04],
       [-0.04]]), 'W2': array([[0.96, 0.96]]), 'b2': array([[-0.04]])}

Cost after iteration 5: 0.5276569599999996

Parameters after iteration 5: {'W1': array([[0.95, 0.95],
       [0.95, 0.95]]), 'b1': array([[-0.05],
       [-0.05]]), 'W2': array([[0.95, 0.95]]), 'b2': array([[-0.05]])}

Cost after iteration 6: 0.4355999999999996

Parameters after iteration 6: {'W1': array([[0.94, 0.94],
       [0.94, 0.94]]), 'b1': array([[-0.06],
       [-0.06]]), 'W2': array([[0.94, 0.94]]), 'b2': array([[-0.06]])}

Cost after iteration 7: 0.3533113599999995

Parameters after iteration 7: {'W1': array([[0.93, 0.93],
       [0.93, 0.93]]), 'b1': array([[-0.07],
       [-0.07]]), 'W2': array([[0.93, 0.93]]), 'b2': array([[-0.07]])}

Cost after iteration 8: 0.2804761599999996

Parameters after iteration 8: {'W1': array([[0.92, 0.92],
```

```
       [0.92, 0.92]]), 'b1': array([[-0.08],
       [-0.08]]), 'W2': array([[0.92, 0.92]]), 'b2': array([[-0.08]])}

Cost after iteration 9: 0.21678335999999981

Parameters after iteration 9: {'W1': array([[0.91, 0.91],
       [0.91, 0.91]]), 'b1': array([[-0.09],
       [-0.09]]), 'W2': array([[0.91, 0.91]]), 'b2': array([[-0.09]])}

Cost after iteration 10: 0.16192575999999972

Parameters after iteration 10: {'W1': array([[0.9, 0.9],
       [0.9, 0.9]]), 'b1': array([[-0.1],
       [-0.1]]), 'W2': array([[0.9, 0.9]]), 'b2': array([[-0.1]])}
```

In [ ]: