ROLLNO:225229105

Deep Learning Lab Sheet 2 - Design of Logic Gates using Perceptron and Keras

Part-I

In [2]:

```
import numpy as np
def logic_gate(w1, w2, b):
    return lambda x1, x2: sigmoid(w1 * x1 + w2 * x2 + b)
def test(gate):
      for a, b in (0, 0), (0, 1), (1, 0), (1, 1):
            print("{}, {}: {}".format(a, b, np.round(gate(a,b))))
```

In [3]:

```
def sigmoid(x):
    return 1/(1+np.exp(-x))
or_gate = logic_gate(20, 20, -10)
test(or_gate)
```

```
0, 0: 0.0
0, 1: 1.0
1, 0: 1.0
1, 1: 1.0
```

Part-II

In [4]:

```
def logic_gate(w1, w2, b):
    return lambda x1, x2: sigmoid(w1 * x1 + w2 * x2 + b)

def test(gate):
        for a, b in [(0, 0), (0, 1), (1, 0), (1, 1)]:
            print("{}, {}: {}".format(a, b, np.round(gate(a, b))))


and_gate = logic_gate(1, 1, -1.5)
print("AND gate:")
test(and_gate)
print()

nor_gate = logic_gate(-1, -1, 0.5)
print("NOR gate:")
test(nor_gate)
print()

nand_gate = logic_gate(-1, -1, 1.5)
print("NAND gate:")
test(nand_gate)
```

```
AND gate:
0, 0: 0.0
0, 1: 0.0
1, 0: 0.0
1, 1: 1.0

NOR gate:
0, 0: 1.0
0, 1: 0.0
1, 0: 0.0
1, 1: 0.0

NAND gate:
0, 0: 1.0
0, 1: 1.0
1, 0: 1.0
1, 1: 0.0
```

Part-III

In [6]:

```
def xor_gate(x1, x2):

    w1 = np.array([[20, -20], [-20, 20]])
    b1 = np.array([-10, 30])
    w2 = np.array([[-20], [-20]])
    b2 = np.array([30])


    a1 = sigmoid(np.dot(np.array([x1, x2]), w1) + b1)


    a2 = sigmoid(np.dot(a1, w2) + b2)

    return a2[0]


print("0 XOR 0 =", xor_gate(0, 0))
print("0 XOR 1 =", xor_gate(0, 1))
print("1 XOR 0 =", xor_gate(1, 0))
print("1 XOR 1 =", xor_gate(1, 1))
```

```
0 XOR 0 = 0.9999545608951235
0 XOR 1 = 0.9999546021312976
1 XOR 0 = 4.548037850511215e-05
1 XOR 1 = 0.9999545608951235
```

Part-IV

In [8]:

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense


input_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
output_data = np.array([[0], [0], [0], [1]])

model = Sequential()
model.add(Dense(16, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['bina
model.fit(input_data, output_data, epochs=100)

predictions = model.predict(input_data)
print(predictions)


input_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
output_data = np.array([[0], [1], [1], [1]])

model = Sequential()
model.add(Dense(16, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['bina
model.fit(input_data, output_data, epochs=100)

predictions = model.predict(input_data)
print(predictions)


input_data = np.array([[0], [1]])
output_data = np.array([[1], [0]])

model = Sequential()
model.add(Dense(16, input_dim=1, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['bina
model.fit(input_data, output_data, epochs=100)

predictions = model.predict(input_data)
print(predictions)


input_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
output_data = np.array([[1], [1], [1], [0]])

model = Sequential()
model.add(Dense(16, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['bina
model.fit(input_data, output_data, epochs=100)

predictions = model.predict(input_data)
print(predictions)
```

```
input_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
output_data = np.array([[1], [0], [0], [0]])

model = Sequential()
model.add(Dense(16, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['bina
model.fit(input_data, output_data, epochs=100)

predictions = model.predict(input_data)
print(predictions)


input_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
output_data = np.array([[0], [1], [1], [0]])

model = Sequential()
model.add(Dense(16, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['bina
model.fit(input_data, output_data, epochs=100)

predictions = model.predict(input_data)
print(predictions)
```

```
Epoch 7/100
1/1 [==============================] - 0s 5ms/step - loss: 0.2584 -
binary_accuracy: 0.5000
Epoch 8/100
1/1 [==============================] - 0s 4ms/step - loss: 0.2577 -
binary_accuracy: 0.5000
Epoch 9/100
1/1 [==============================] - 0s 8ms/step - loss: 0.2571 -
binary_accuracy: 0.5000
Epoch 10/100
1/1 [==============================] - 0s 4ms/step - loss: 0.2564 -
binary_accuracy: 0.5000
Epoch 11/100
1/1 [==============================] - 0s 7ms/step - loss: 0.2558 -
binary_accuracy: 0.5000
Epoch 12/100
1/1 [==============================] - 0s 4ms/step - loss: 0.2551 -
binary_accuracy: 0.5000
Epoch 13/100
1/1 [==============================] - 0s 6ms/step - loss: 0.2545 -
```

In [ ]:

```
########################################################################
```