Lab 3

November 25, 2021

Lab 3: Neural Network to simulate an OR gate using Hebian Learning

Hebbian Learning Rule, also known as Hebb Learning Rule, was proposed by Donald Hebb. It is one of the first and also easiest learning rules in the neural network. It is a single layer neural network, i.e. it has one input layer and one output layer. The input layer can have many units, say n. The output layer only has one unit. Hebbian rule works by updating the weights between neurons in the neural network for each training sample.

Learning Rule

Step 0: initialize all weights to 0

Step 1: Given a training input, s, with its target output, t, set the activation of the input units: xi = si

Step 2: Set the activation of the output unit to the target value: y = t

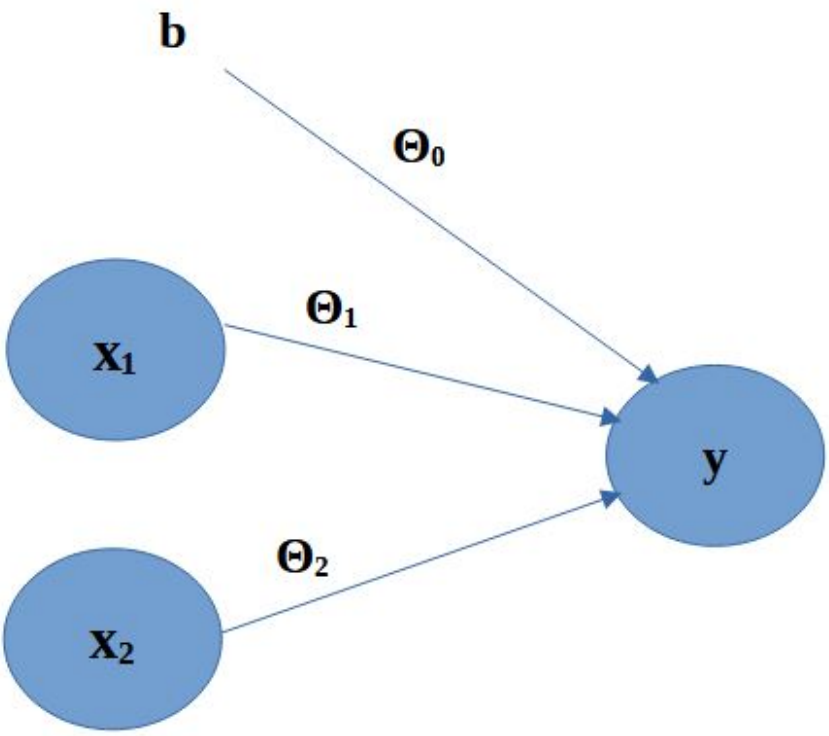
Step 3: Adjust the weights: wi (new) = wi (old) + xi y

Step 4: Adjust the bias (just like the weights): b(new) = b(old) + y

We are going to learn weights of NN for following set of inputs and outputs using hebian learning.

bias	x1	x2	y
1	-1	-1	-1
1	-1	1	1
1	1	-1	1
1	1	1	1

The structure of network is shown below



[1]:
```
#we import the numpy library
import numpy as np
```

[2]:
```
#Initialize x1, x2 and y
b = [1, 1, 1, 1]
x1 = [1, 1, -1, -1]
x2 = [1, -1, 1, -1]
y = [1, 1, 1, -1]
```

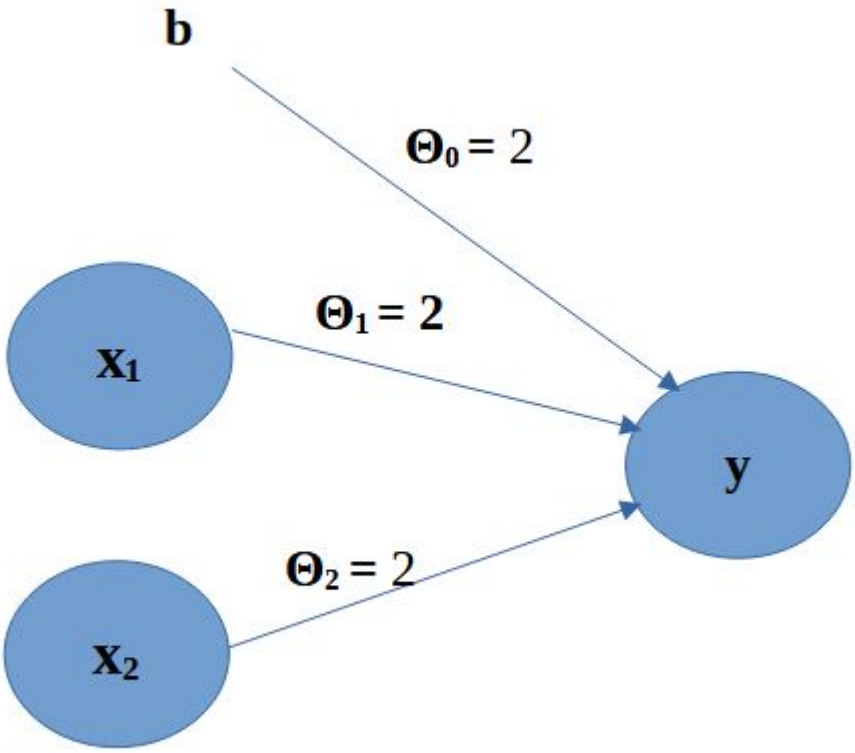
[3]:
```
#Initialize weights to 0
theta = [0, 0, 0]
```

[4]:
```
#Use Hebian Learning Rule
for i in range(0,len(x1)):
    theta[1] = theta[1] + x1[i]*y[i]
    theta[2] = theta[2] + x2[i]*y[i]
    theta[0] = theta[0] + y[i]
```

[5]:
```
theta
```

[5]: `[2, 2, 2]`

The structure of network with the weights after training is shown below



[6]:
```
#we define the sigmoid function
def sigmoid(Z):
    return 1/(1 + np.exp(-Z))
```

[7]:
```
def predict(X,theta):
    Y = X[1]*theta[1] + X[2]*theta[2] + theta[0]
    print("The Sigmoid is",sigmoid(Y))
    if sigmoid(Y) >= 0.5:
        return 1
    else:
        return -1
```

[12]:
```
X = np.array([1,-1,-1])
print("The Prediction is ",predict(X,theta))
```

```
The Sigmoid is 0.11920292202211755
The Prediction is  -1
```

[13]:
```
X = np.array([1,-1,1])
print("The Prediction is ",predict(X,theta))
```

```
The Sigmoid is 0.8807970779778823
The Prediction is  1
```

[14]:
```
X = np.array([1,1,-1])
print("The Prediction is ",predict(X,theta))
```

```
The Sigmoid is 0.8807970779778823
The Prediction is  1
```

[15]:
```
X = np.array([1,1,1])
print("The Prediction is ",predict(X,theta))
```

```
The Sigmoid is 0.9975273768433653
The Prediction is  1
```