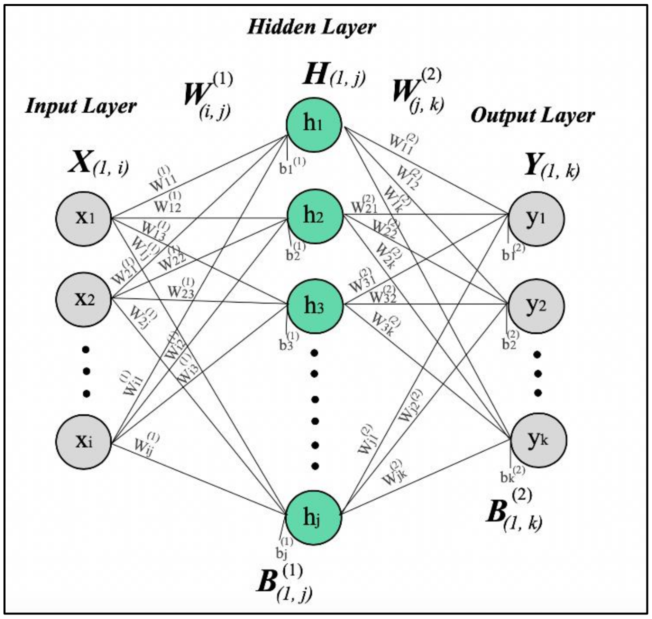
# FEED FORWARD MULTILAYER NEURAL NETWORK

XOR with backpropagation
Handwritten digits recognition
Handwritten alphabet recognition

SRI SAI VIJAYA ADITYA NITTALA

ROLL NUMBER: 177163
REGISTRATION NUMBER: 841775

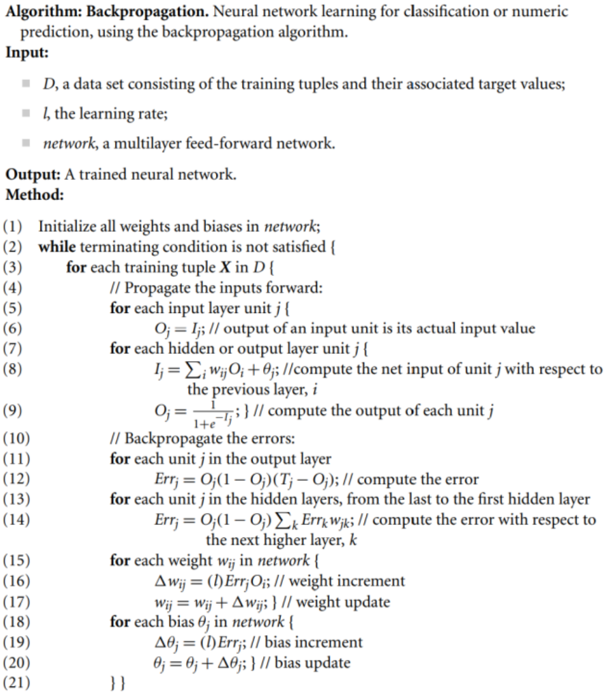
The architecture of a 3-layer Neural Network is shown below:



Where,

- X: Input features fed into the Neural Network.
- W: Weights associated with the activations of the Neural Network.
- B: Bias term introduced to the Neural Network.

A Neural Network can be used for classification purposes. The algorithm that is used to classify is called ***Backpropagation.*** An overview is given below:



**Algorithm: Backpropagation.** Neural network learning for classification or numeric prediction, using the backpropagation algorithm.

**Input:**

- $D$, a data set consisting of the training tuples and their associated target values;
- $l$, the learning rate;
- *network*, a multilayer feed-forward network.

**Output:** A trained neural network.

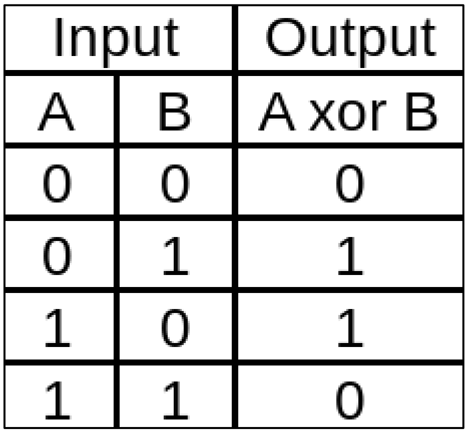
**Method:**

```
Initialize all weights and biases in network;
while terminating condition is not satisfied {
    for each training tuple $X$ in $D$ {
            // Propagate the inputs forward:
            for each input layer unit $j$ {
                    $O_j = I_j$; // output of an input unit is its actual input value
            for each hidden or output layer unit $j$ {
                    $I_j = \sum_i w_{ij} O_i + \theta_j$; //compute the net input of unit $j$ with respect to
                         the previous layer, $i$
                    $O_j = \frac{1}{1+e^{-I_j}}$; } // compute the output of each unit $j$
            // Backpropagate the errors:
            for each unit $j$ in the output layer
                    $Err_j = O_j(1 - O_j)(T_j - O_j)$; // compute the error
            for each unit $j$ in the hidden layers, from the last to the first hidden layer
                    $Err_j = O_j(1 - O_j)\sum_k Err_k w_{jk}$; // compute the error with respect to
                         the next higher layer, $k$
            for each weight $w_{ij}$ in network {
                    $\Delta w_{ij} = (l) Err_j O_i$; // weight increment
                    $w_{ij} = w_{ij} + \Delta w_{ij}$; } // weight update
            for each bias $\theta_j$ in network {
                    $\Delta \theta_j = (l) Err_j$; // bias increment
                    $\theta_j = \theta_j + \Delta \theta_j$; } // bias update
            } }
```

In this assignment, using the ***backpropagation*** algorithm, the following have been implemented:

1. **<u>Exclusive OR (XOR) gate implementation</u>:**

The truth-table of an XOR gate is given below:

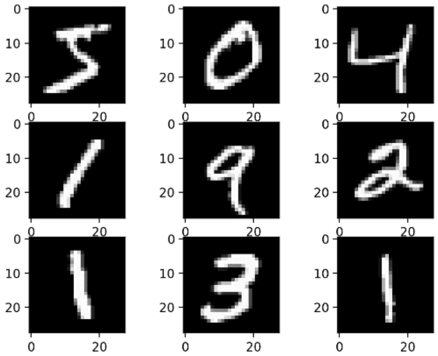


| Input | | Output |
|---|---|---|
| A | B | A xor B |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Information about the implementation:**

a. Number of neurons in input layer: 2
b. Number of hidden layers: 1
c. Number of neurons in the hidden layer: 2
d. Number of neurons in output layer: 1
e. Input: Dataset is essentially the truth-table of XOR but 0s are represented by 0.1 and 1s with 0.9 as sigmoid cannot compute exactly 0 or 1.
f. Number of iterations: 1000
g. Learning rate: 0.1
h. Weights are update after each forward-backward pass, i.e, not a batch update.
i. Test data: truth-table of XOR

2. **<u>Handwritten Digit recognition</u>:**

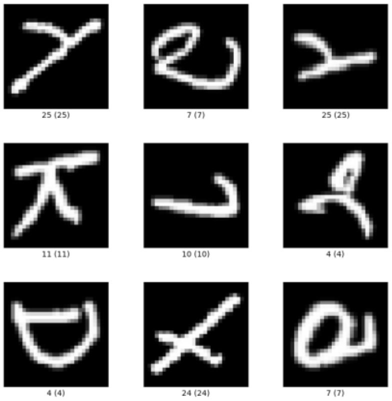


The dataset used for this application is the **MNIST** dataset. It consists of images, each of size **28x28** pixels (total 784 per image). Number of training and testing examples taken is less due to infrastructure constraints. Code attached at the end contains more information about the input.

**Information about the implementation:**

a. Number of neurons in input layer: 784
b. Number of hidden layers: 1
c. Number of neurons in hidden layer: 50
d. Number of neurons in output layer: 10
e. Input:
    a. Number of training examples considered: 1000
    b. Number of testing examples considered: 100
f. Learning rate: 0.01
g. Number of iterations: 10

3. **Handwritten alphabet recognition**:



The dataset used for this application is the **EMNIST** dataset. It is very similar to the previously described MNIST dataset. Contains images of size **28x28**, total of 784 pixels. More information is provided as comments in the code attached below.

**Information about the implementation:**

a. Number of neurons in input layer: 784
b. Number of hidden layers: 1
c. Number of neurons in hidden layer: 50
d. Number of neurons in output layer: 26
e. Input:
    a. Number of training examples considered: 2400
    b. Number of testing examples considered: 600
f. Learning rate: 0.01
g. Number of iterations: 10