# Neural Network

Introduction

Khaleda  Akther Papry

Assistant Professor,CSE,DUET

Email: papry.khaleda@duet.ac.bd

Room: 7023 (New academic building)

## Course Overview

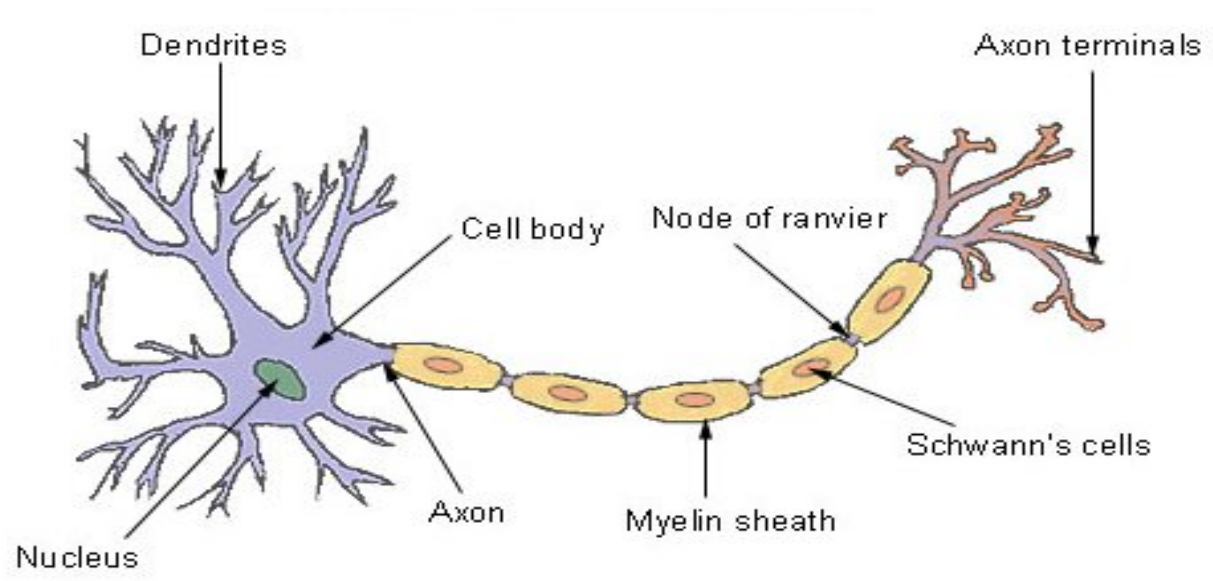
- Class time : Wednesday 2:00p.m-4:30p.m
- Reference Book: Fundamentals of Neural Networks; Architectures, Algorithms, and Applications – Laurance Fausett
- Marks Distribution (Tentative)
    - Attendance  - 10%
    - Mid term exam - 20%
    - Project - 40% (Presentation + Report)
    - Term Final - 30%

## Introduction

- What is a Neural Network?
  - Neural Networks (NN), also called as Artificial Neural Network is named after its artificial representation of working of a human being’s nervous system.

## How Human brain (nervous system)works?

- Nervous System comprises of millions of nerve cells or neurons. A neuron has the following structure:

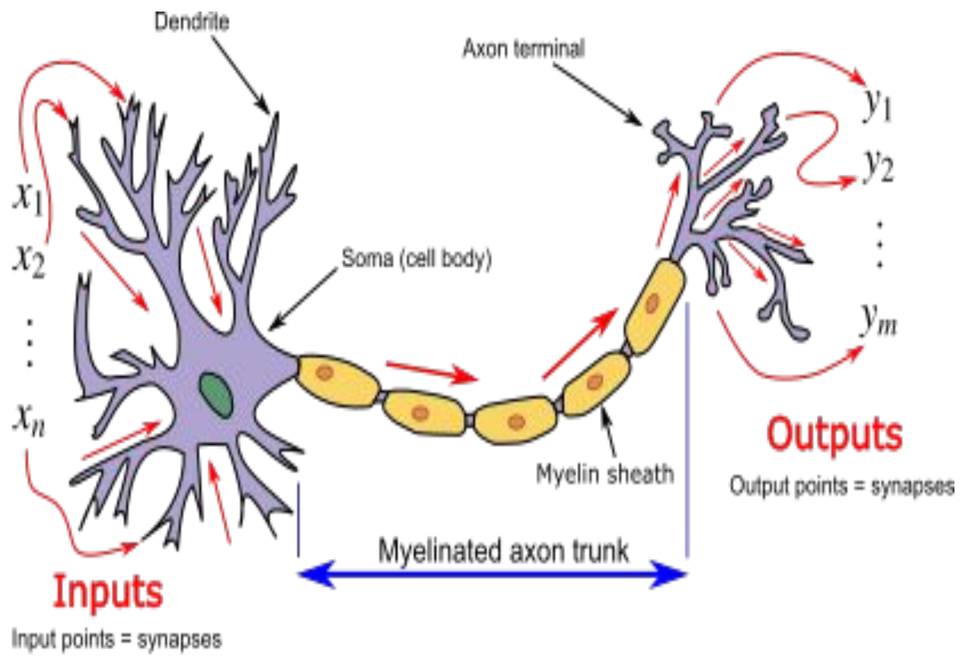


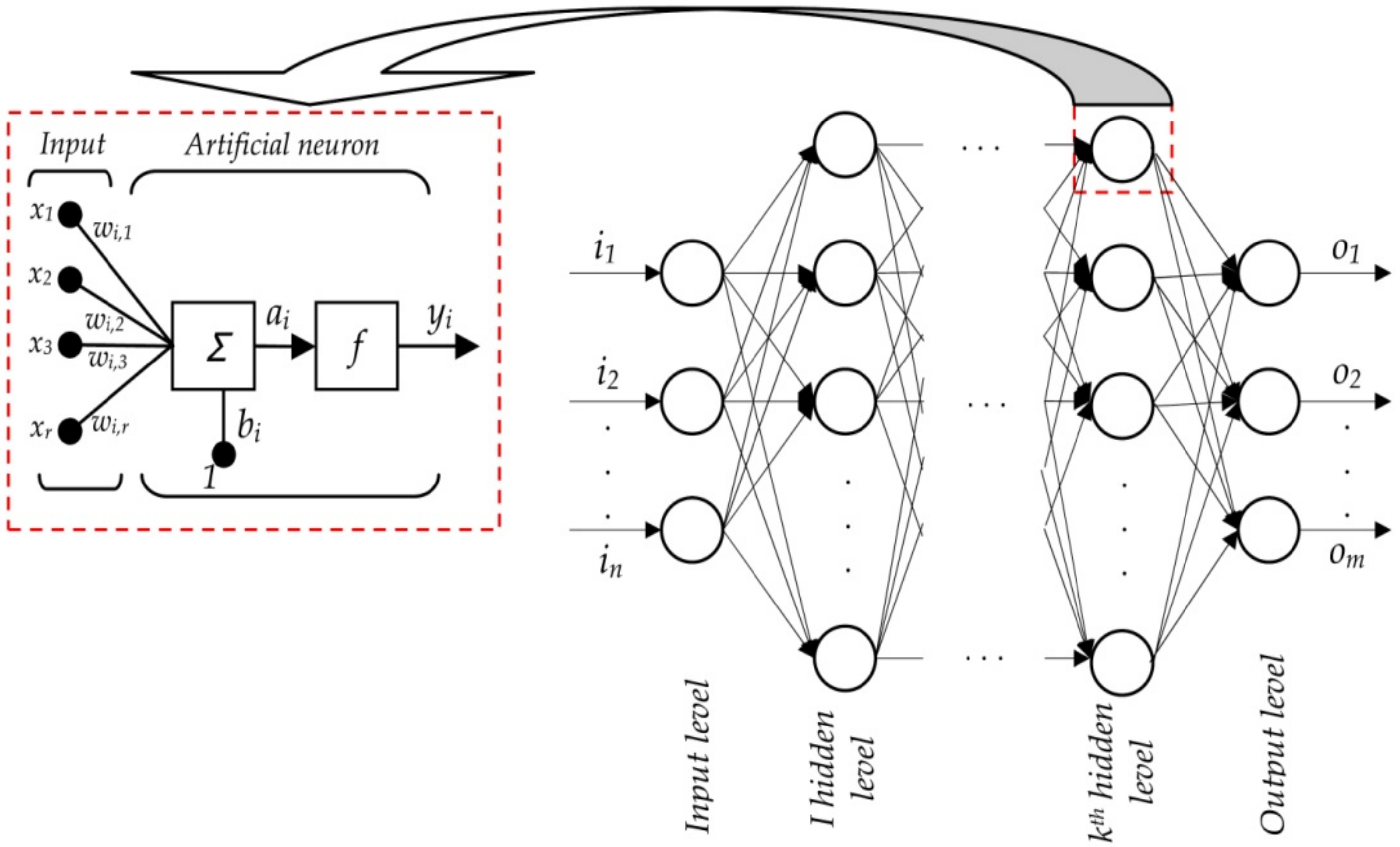
## *H*ow Human brain (nervous system)works?

- The major components are:
- **Dendrites-** It takes input from other neurons in form of an electrical impulse
- **Cell Body**– It generate inferences from those inputs and decide what action to take
- **Axon terminals**– It transmit outputs in form of electrical impulse
- In simple terms, each neuron takes input from numerous other neurons through the dendrites. It then performs the required processing on the input and sends another electrical pulse through the axiom into the terminal nodes from where it is transmitted to numerous other neurons.

## Artificial Neural Network (ANN)

- ANN works in a very similar fashion. The general structure of a neural network looks like:

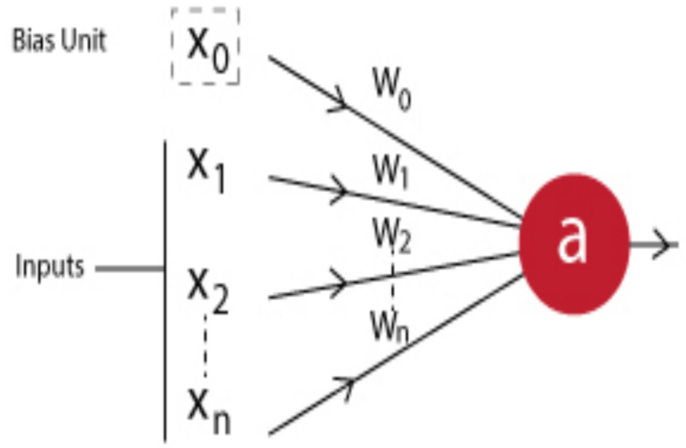




# How a Single Neuron works?

- In this section, we will explore the working of a single neuron with easy examples. The idea is to give you some intuition on how a neuron compute outputs using the inputs. A typical neuron looks like:

- The different components are:
  - **$x_1, x_2, \ldots, x_N$:** Inputs to the neuron. These can either be the actual observations from input layer or an intermediate value from one of the hidden layers.
  - **$x_0$:** Bias unit. This is a constant value added to the input of the activation function. It works similar to an intercept term and typically has +1 value.
  - **$w_0, w_1, w_2, \ldots, w_N$:** Weights on each input. Note that even bias unit has a weight.
  - **a:** Output of the neuron which is calculated as:



$$a = f\left(\sum_{i=0}^{N} w_i x_i\right)$$

Here **f** is known an **activation function**. It can be a gaussian function, logistic function, hyperbolic function or even a linear function in simple cases.

## NN working principle

- Lets implement 3 fundamental functions –
  - OR, AND, NOT using Neural Networks.
- This will help us understand how they work. You can assume these to be like a classification problem where we'll predict the output (0 or 1) for different combination of inputs.
- We will model these like linear classifiers with the following activation function:

$$f(x) = \begin{cases} 0, & for\; x < 0 \\ 1, & for\; x \geq 0 \end{cases}$$

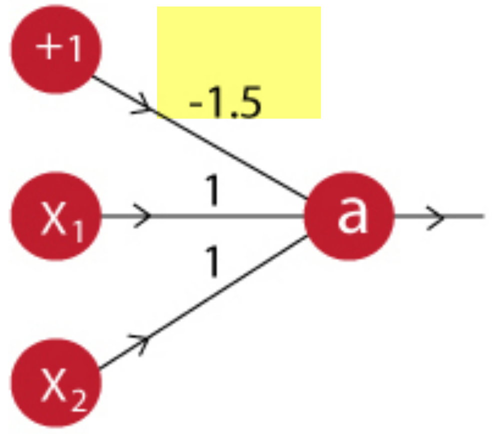
## Example: AND function

- The AND function can be implemented as:

- The output of this neuron is:

  a = f( -1.5 + x1 + x2 )

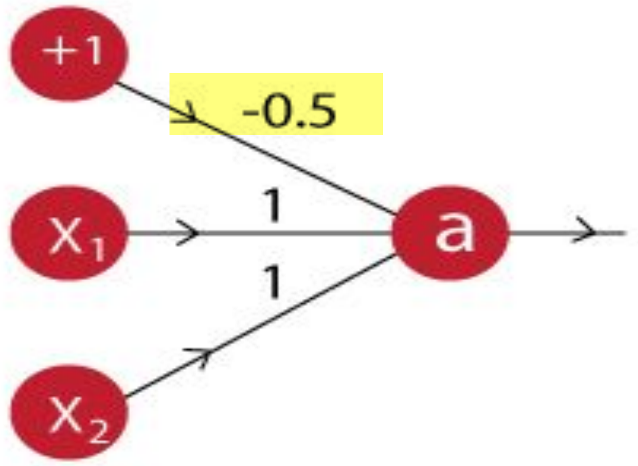
- The truth table for this implementation is:



| X1 | X2 | X1 AND X2 | (-1.5+X1+X2) | a |
|---|---|---|---|---|
| 0 | 0 | 0 | -1.5 | 0 |
| 0 | 1 | 0 | -0.5 | 0 |
| 1 | 0 | 0 | -0.5 | 0 |
| 1 | 1 | 1 | 0.5 | 1 |

# Example: OR function

- The OR function can be implemented as:



- The output of this neuron is:

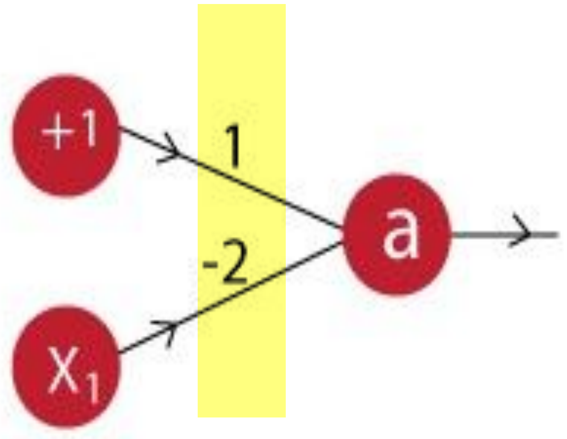
  a = f( -0.5 + x1 + x2 )

- The truth table for this implementation is:

| X1 | X2 | X1 OR X2 | (-0.5+X1+X2) | a |
|---|---|---|---|---|
| 0 | 0 | 0 | -0.5 | 0 |
| 0 | 1 | 1 | 0.5 | 1 |
| 1 | 0 | 1 | 0.5 | 1 |
| 1 | 1 | 1 | 1.5 | 1 |

# Example: NOT function

- the NOT function can be implemented as:



- The output of this neuron is:

a = f( 1 $\boxed{- 2*x1 )}$

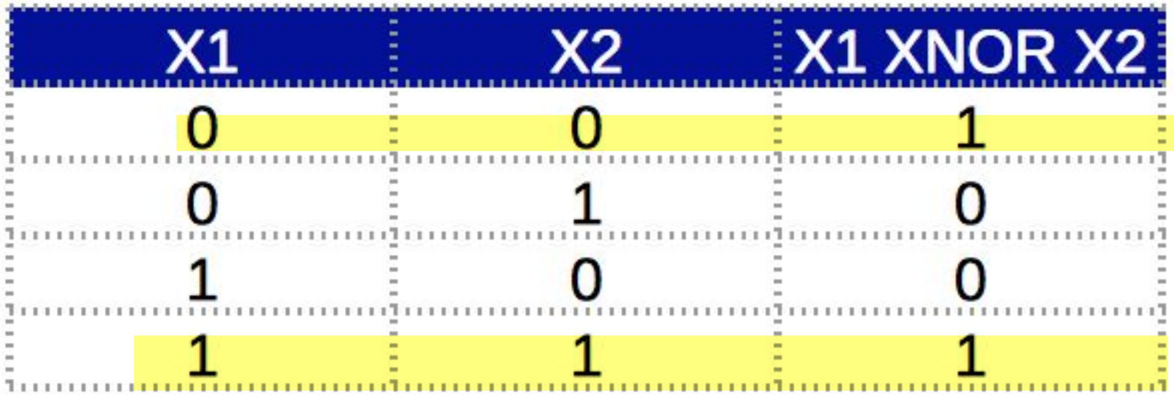
f(1+(-2*1))

- The truth table for this implementation is:

| X1 | NOT X1 | (1-2*X1) | a |
|---|---|---|---|
| 0 | 1 | 1 | 1 |
| 1 | 0 | -1 | 0 |

## Why multi-layer networks are useful?

- The example of an **XNOR** function. Just a recap, the truth table of an **XNOR** function looks like:



| X1 | X2 | X1 XNOR X2 |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

- Here we can see that the output is 1 when both inputs are same, otherwise 0. This sort of a relationship cannot be modeled using a single neuron.

# Why multi-layer networks are useful?

- The idea behind using multiple layers is that complex relations can be broken into simpler functions and combined.
- Lets break down the **XNOR** function.

X1 **XNOR** X2 = **NOT** ( X1 **XOR** X2 )

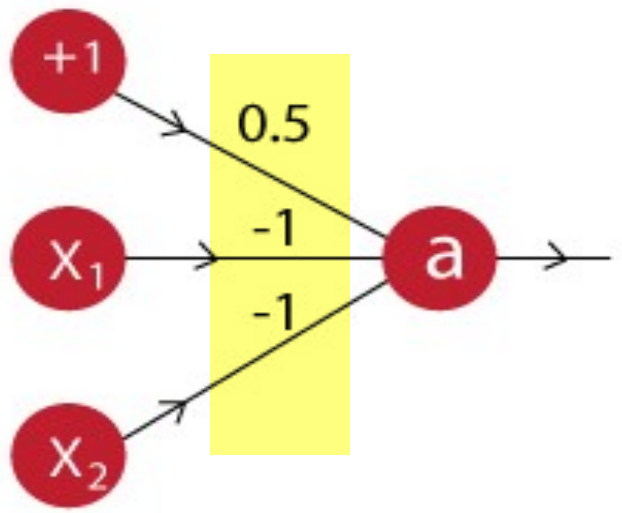
= **NOT** [ (A+B).(A'+B') ]

= (A+B)' + (A'+B')'

= (A'.B') + (A.B)

# Example 4: XNOR Function

- Design a neuron to model **A'.B'.**
- This can be easily modeled using the following:



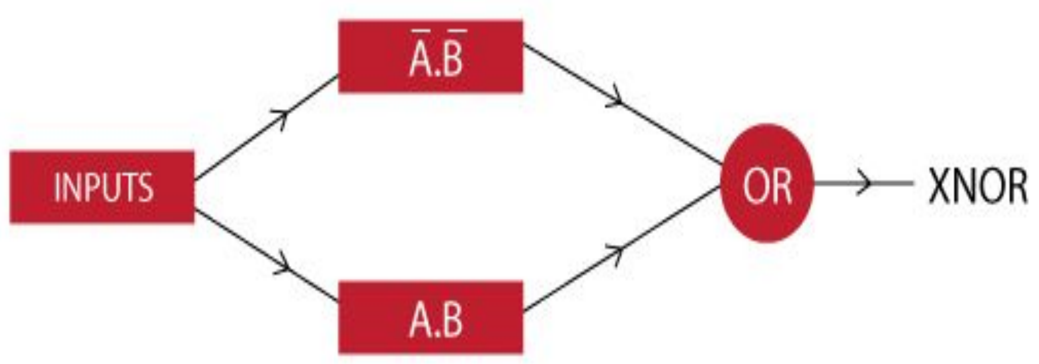
- The output of this neuron is:

  a = f( 0.5 – x1 – x2 )
- The truth table for this function is

| X1 | X2 | X1' AND X2' | (0.5-X1-X2) | a |
|---|---|---|---|---|
| 0 | 0 | 1 | 0.5 | 1 |
| 0 | 1 | 0 | -0.5 | 0 |
| 1 | 0 | 0 | -0.5 | 0 |
| 1 | 1 | 0 | -1.5 | 0 |

## Example 4: XNOR Function

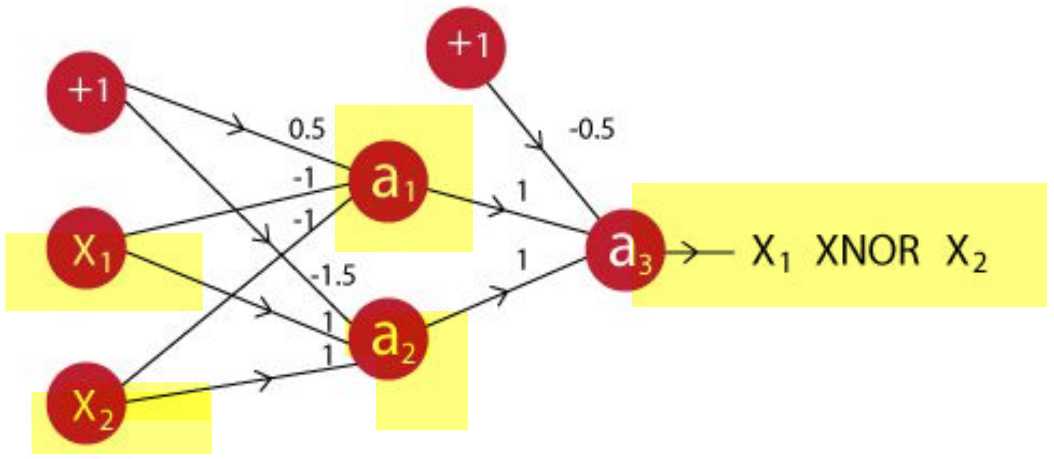
- First, lets look at the semantic diagram of that network:

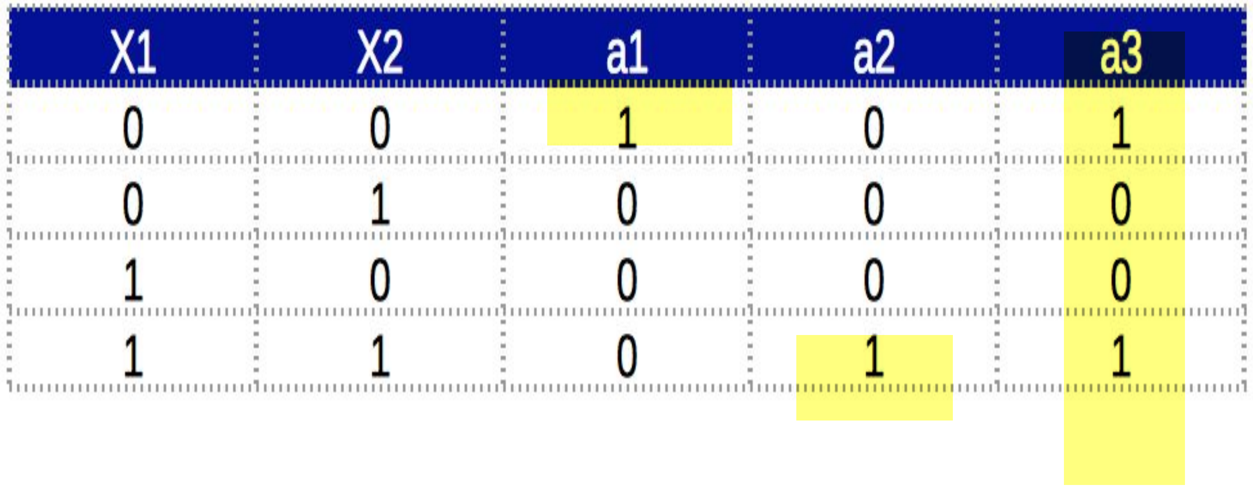


- In layer 1, we will determine A'.B' and A.B individually.
- In layer 2, we will take their output and implement an OR function on top.

# Example 4: XNOR Function

- The final network would look like this:





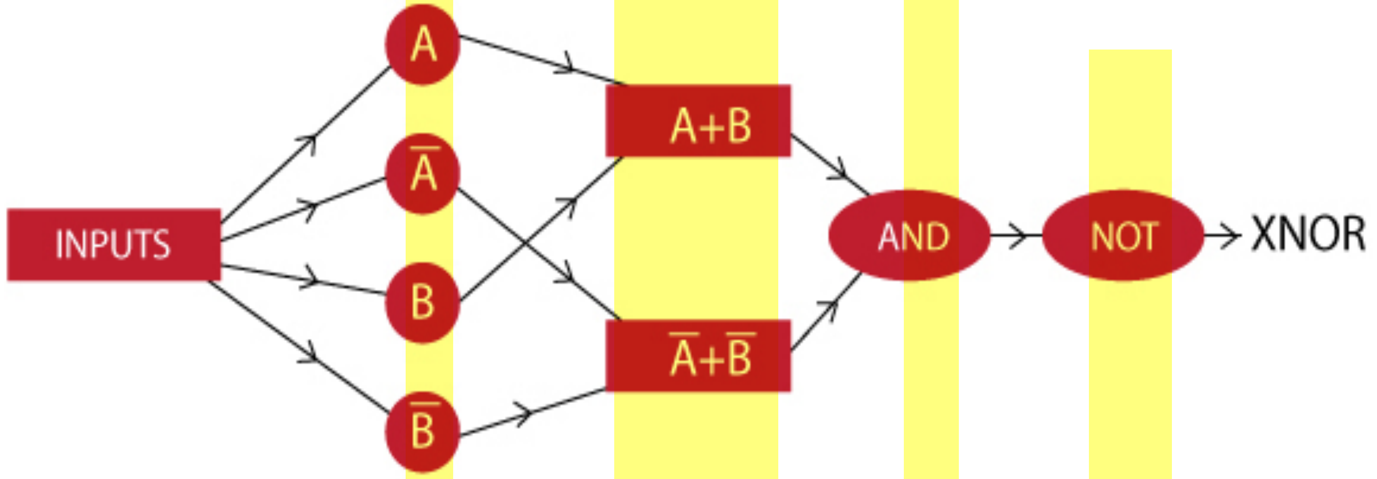
| X1 | X2 | a1 | a2 | a3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |

The different outputs represent different units:

1. **$a_1$**: implements A’.B’
2. **$a_2$**: implements A.B
3. **$a_3$**: implements OR which works on a1 and a2, thus effectively (A’.B’ + A.B)

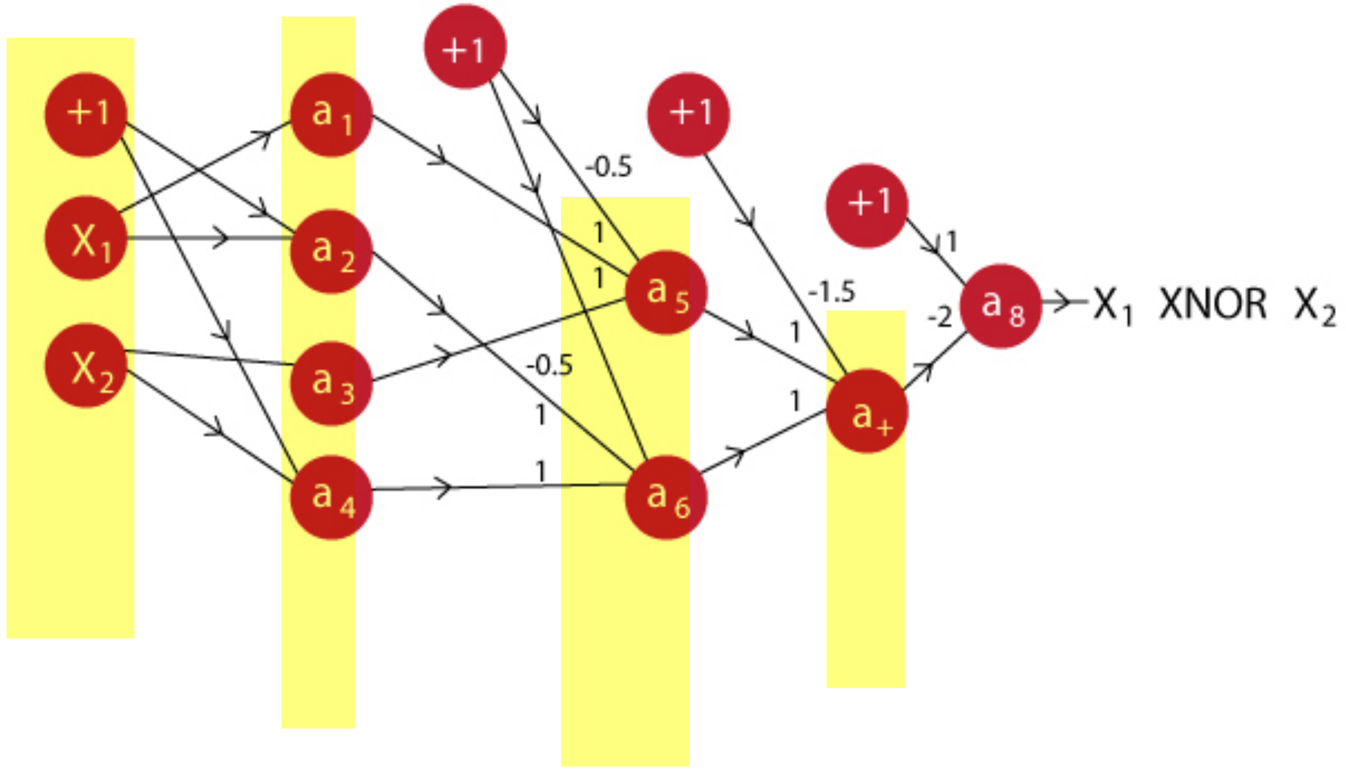
# Another implementation of XNOR Function

- X1 XNOR X2 = NOT [ (A+B).(A'+B') ]
- Consider the following semantic:



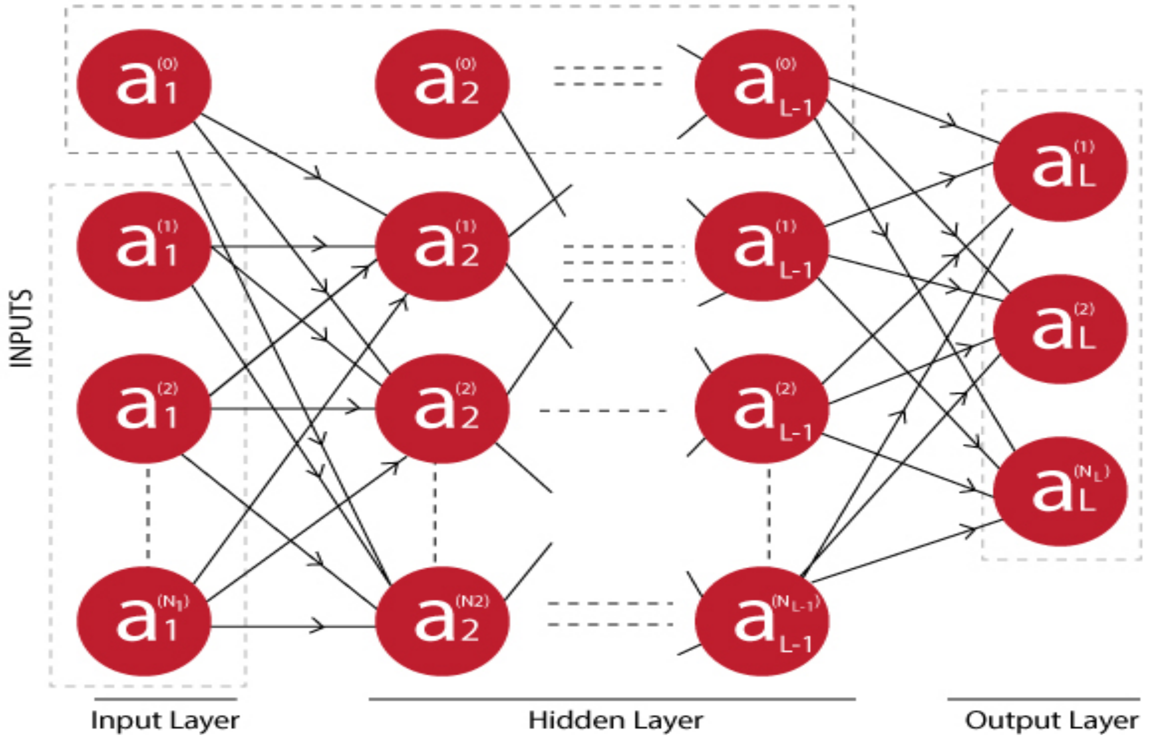
# Another implementation of XNOR Function

- We had to use 3 hidden layers.
- The network looks like:



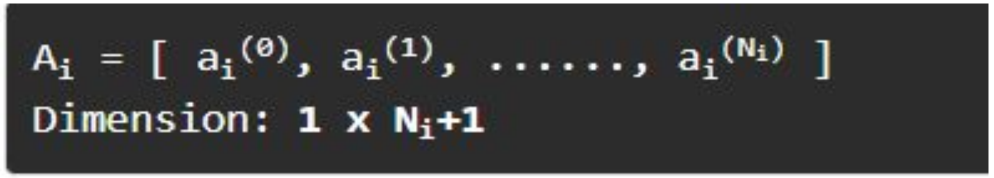
# General Structure of a Neural Network

- It has L layers with 1 input layer, 1 output layer and L-2 hidden layers.
- Terminology:
  - **L:** number of layers
  - **$N_i$:** number of neuron in ith layer excluding the bias unit, where i=1,2,…,L
  - **$a_i^{(j)}$:** the output of the jth neuron in ith layer, where i=1,2…L | j=0,1,2….$N_i$



# General Structure of a Neural Network

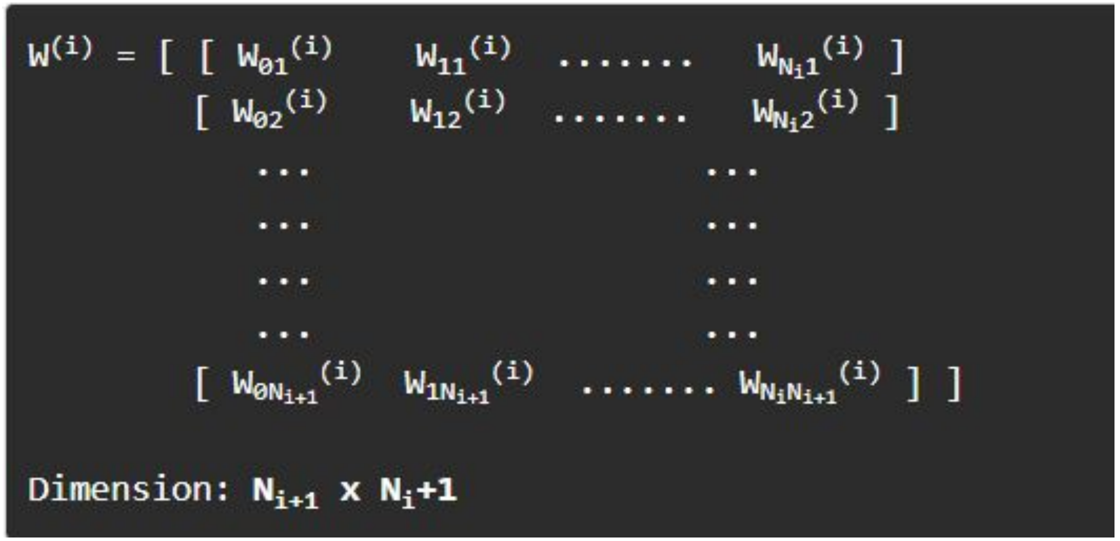
- The output of each layer forms the input of next layer
- The output of i+1-th layer using output of i-th layer as input.
- The input to the i+1-th layer are:



$$A_i = [\ a_i^{(0)},\ a_i^{(1)},\ \ldots\ldots,\ a_i^{(N_i)}\ ]$$

Dimension: $\mathbf{1 \times N_i{+}1}$

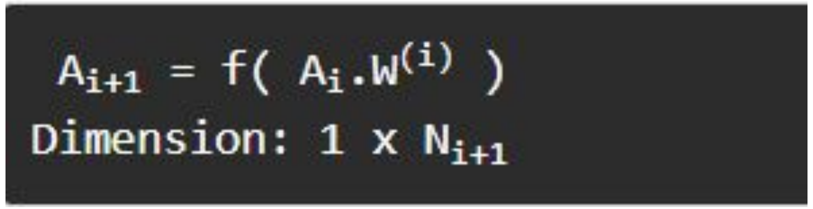
- The weights matrix from i-th to i+1-th layer is:



$$W^{(i)} = \begin{bmatrix} [\ W_{01}^{(i)} & W_{11}^{(i)} & \ldots\ldots & W_{N_i1}^{(i)}\ ] \\ [\ W_{02}^{(i)} & W_{12}^{(i)} & \ldots\ldots & W_{N_i2}^{(i)}\ ] \\ \ldots & & \ldots & \\ \ldots & & \ldots & \\ \ldots & & \ldots & \\ \ldots & & \ldots & \\ [\ W_{0N_{i+1}}^{(i)} & W_{1N_{i+1}}^{(i)} & \ldots\ldots & W_{N_iN_{i+1}}^{(i)}\ ] \end{bmatrix}$$

Dimension: $\mathbf{N_{i+1} \times N_i{+}1}$

- The output of the i+1th layer can be calculated as:



$$A_{i+1} = f(\ A_i . W^{(i)}\ )$$

Dimension: $1 \times N_{i+1}$

## THE END