ANN AND DEEP LEARNING (CS636)

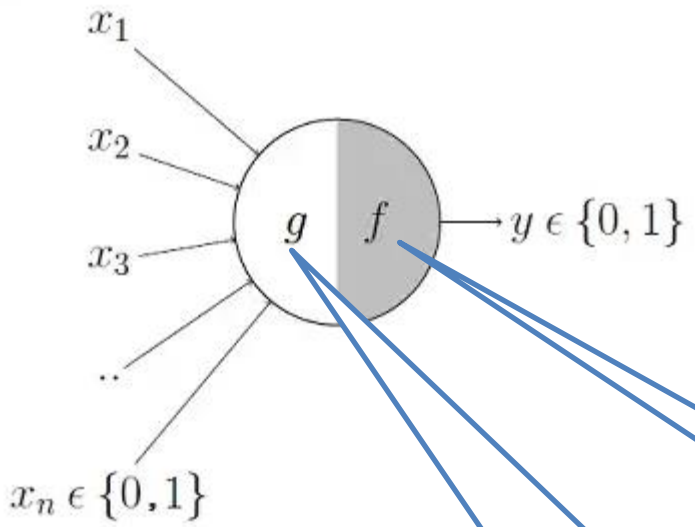
BY: Nidhi S. Periwal,
Teaching Assistant,
COED, SVNIT, Surat

Fundamentals of NN

(Cont..)

MuCulloch & Pitts Neuron

- The first computational model of a neuron was proposed by Warren MuCulloch (neuroscientist) and Walter Pitts (logician) in 1943.
- Idea of neural networks as **computing machines**.



$$g(x_1, x_2, x_3, ..., x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

$$y = f(g(\mathbf{x})) = 1 \quad if \quad g(\mathbf{x}) \geq \theta$$
$$= 0 \quad if \quad g(\mathbf{x}) < \theta$$

g: Summation/Aggregation Function

f: Threshold -> for making decisions

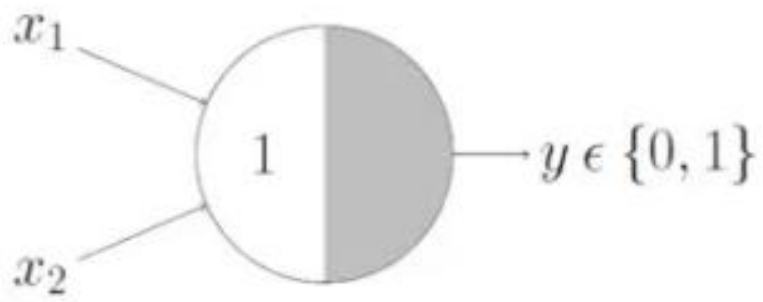
MuCulloch & Pitts Neuron

- ***Excitatory*** *Signals are those that prompt* ***one neuron to share information with the next through an action potential****, while* ***inhibitory*** *signals* ***reduce*** *the probability that such a transfer will take place.*
- Inputs in MP Neuron can either be ***excitatory*** **or** ***inhibitory***.

Inputs Example

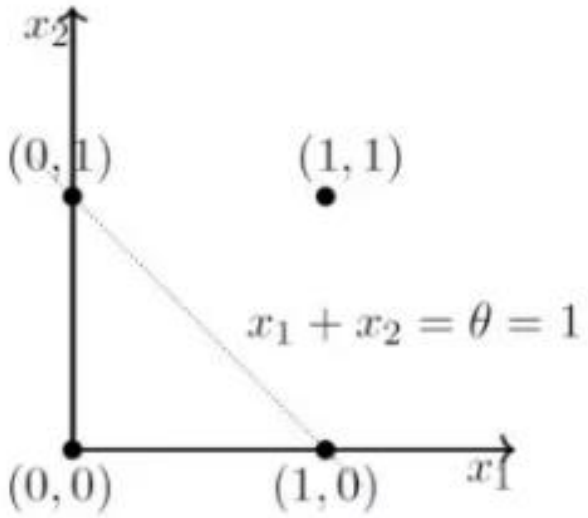
- John will play cricket or not?
- Inputs:
  - X_1 -> AlongwithFriend ? -> Yes/No
  - X_2 -> AtCricketGround ? -> Yes/No
  - X_3 -> AtHome? -> Yes/No
- X_3 as inhibitory
- **Inhibitory inputs** are those that have **maximum effect** on the decision making regardless of other inputs
- **Excitatory inputs** are NOT the ones that will make the neuron fire on their own but they might fire it when combined together.

Boolean Functions Using MP Neuron



OR function

$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 1$$

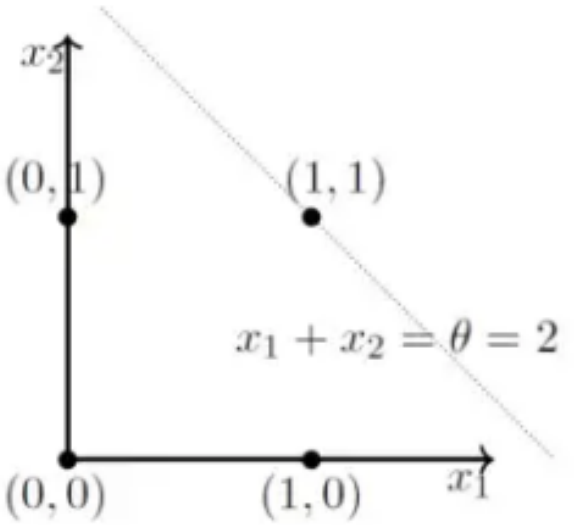


Boolean Functions Using MP Neuron

AND Function

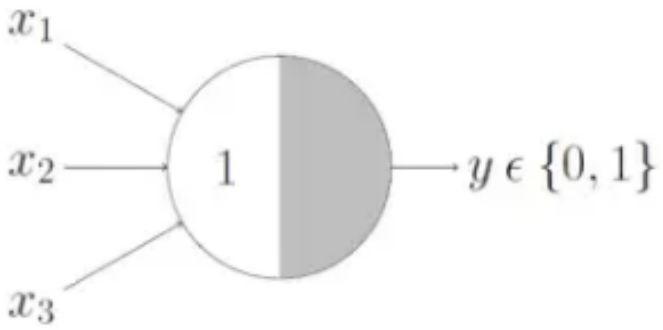
$AND\ function$

$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 2$$



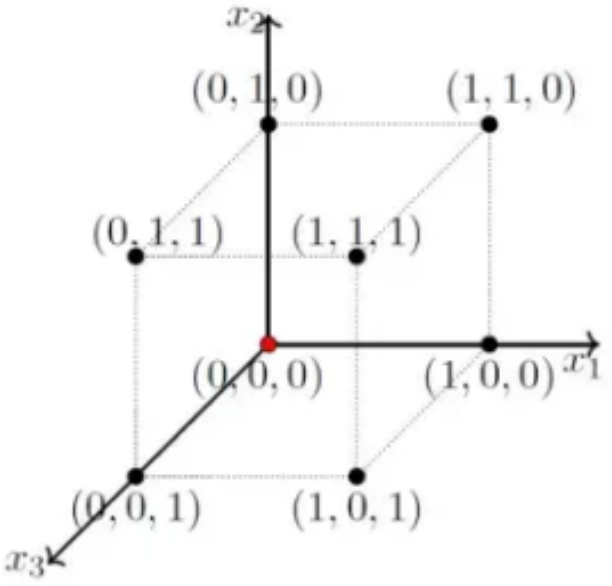
Boolean Functions Using MP Neuron

OR Function With 3 Inputs



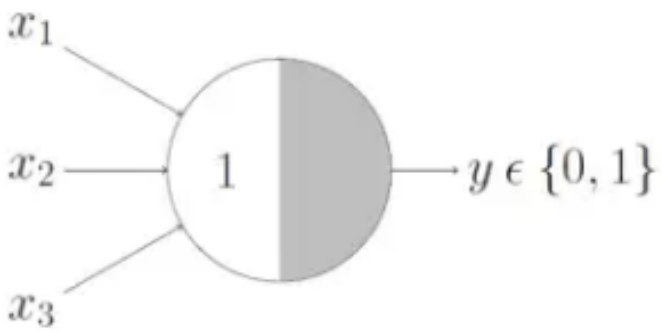
OR function

$$x_1 + x_2 + x_3 = \sum_{i=1}^{3} x_i \geq 1$$



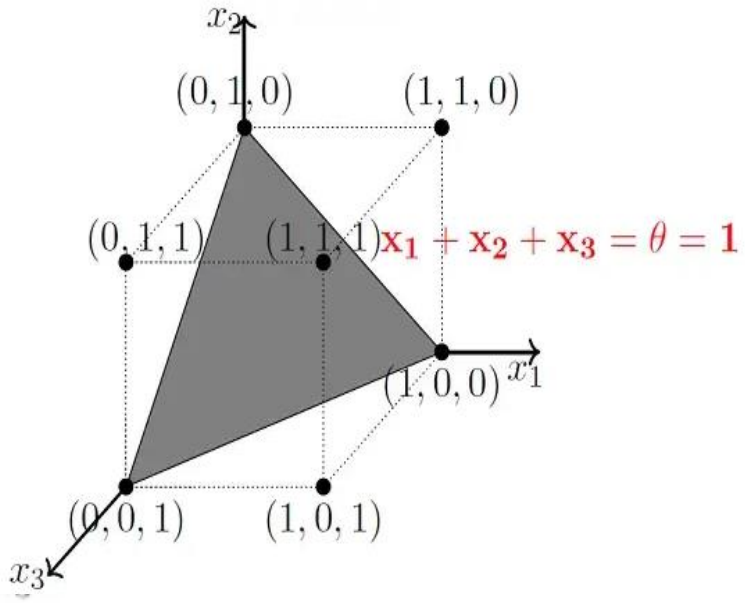
Boolean Functions Using MP Neuron

OR Function With 3 Inputs



OR function

$$x_1 + x_2 + x_3 = \sum_{i=1}^{3} x_i \geq 1$$



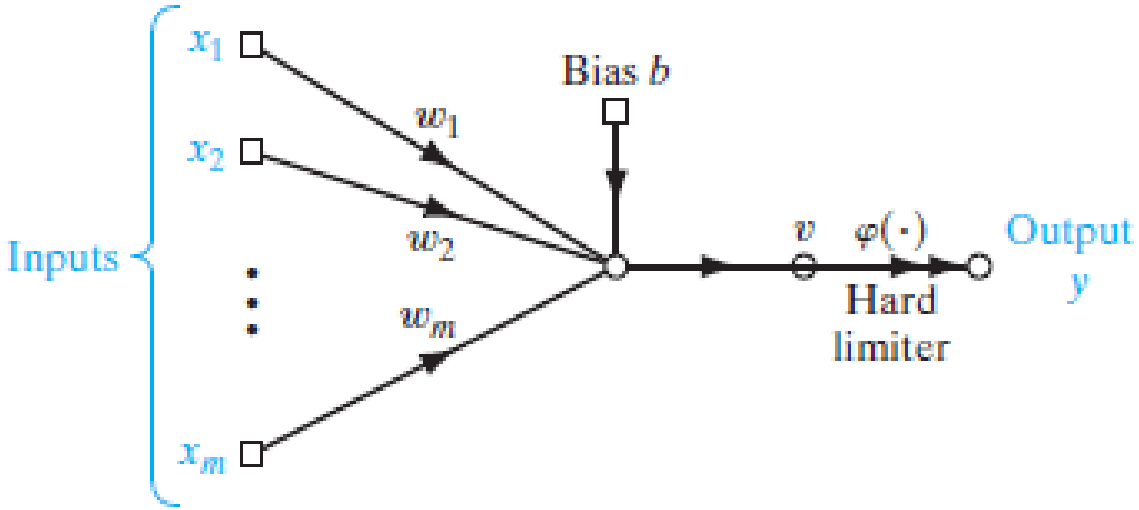
MuCulloch & Pitts Neuron Model

- Boolean Inputs
  - i.e. $X_1, X_2, \ldots X_n$ are always 0/1
- Inputs are not Weighted
  - i.e. Consider $w_1, w_2, \ldots w_n$ always as 1
- MP Neuron Model works linearly separable data.
- Model can represent Boolean function which are linearly separable, where a line/plane can be used to separate data.
- However, we can adjust threshold input to make the model fit our the dataset

Rosenblatt's Perceptron

- Rosenblatt (1958) proposed perceptron as the first model supervised learning.
- The perceptron is the **simplest form** of a neural network used for the classification of patterns said to be *linearly separable.*
- It consists of a single neuron with **adjustable synaptic weights and bias.**
- Rosenblatt proved that if the patterns (vectors) used to train the perceptron are drawn from two linearly separable classes, then the perceptron algorithm converges and positions the decision surface in the form of a hyperplane between the two classes.

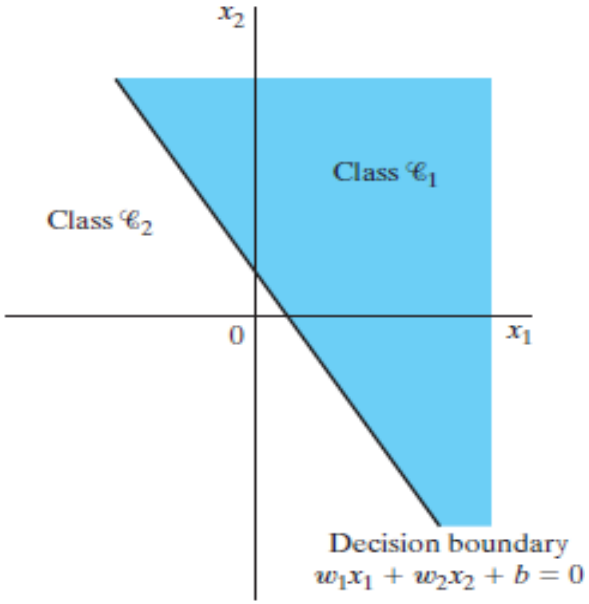
Rosenblatt's Perceptron



$$v = \sum_{i=1}^{m} w_i x_i + b$$

- The summing node of the neural model computes a linear combination of the inputs applied to its synapses, as well as incorporates an externally applied bias.
- The resulting sum, that is, the induced local field, is applied to a hard limiter.
- The neuron produces an output equal to 1 if the hard limiter input is positive, and -1 if it is negative.

Rosenblatt's Perceptron



- Fig shows decision boundary for a two-dimensional, two-class pattern-classification problem.
- The synaptic weights $w_1, w_2, \ldots, w_n$ of the perceptron can be adapted on an iteration by- iteration basis. For the adaptation, we may use an error-correction rule known as the perceptron convergence algorithm

Rosenblatt's Perceptron Model

- Work on linearly separable data
- Numeric Weights (Weighted Input-i.e. All inputs are not equal)
- Real inputs (Not restricted to Boolean)

Perceptron Learning Algorithm

$P \leftarrow$ *inputs with label* 1;
$N \leftarrow$ *inputs with label* 0;
Initialize $\mathbf{w} = [w_1, w_2, \ldots, w_N]$ randomly;
while !*convergence* **do**
  Pick random $\mathbf{x} \in P \cup N$;
  if $\mathbf{x} \in P$ *and* $\sum_{i=0}^{n} w_i * x_i < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
  end
  if $\mathbf{x} \in N$ *and* $\sum_{i=0}^{n} w_i * x_i \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
  end
end
//the algorithm converges when all the inputs are classified correctly

- Consider two vectors $\mathbf{w}$ and $\mathbf{x}$

$$\mathbf{w} = [w_0, w_1, w_2, ..., w_n]$$
$$\mathbf{x} = [1, x_1, x_2, ..., x_n]$$
$$\mathbf{w} \cdot \mathbf{x} = \mathbf{w}^{\mathbf{T}}\mathbf{x} = \sum_{i=0}^{n} w_i * x_i$$

- We can thus rewrite the perceptron rule as

$$y = 1 \quad if \quad \mathbf{w}^{\mathbf{T}}\mathbf{x} \geq 0$$
$$= 0 \quad if \quad \mathbf{w}^{\mathbf{T}}\mathbf{x} < 0$$

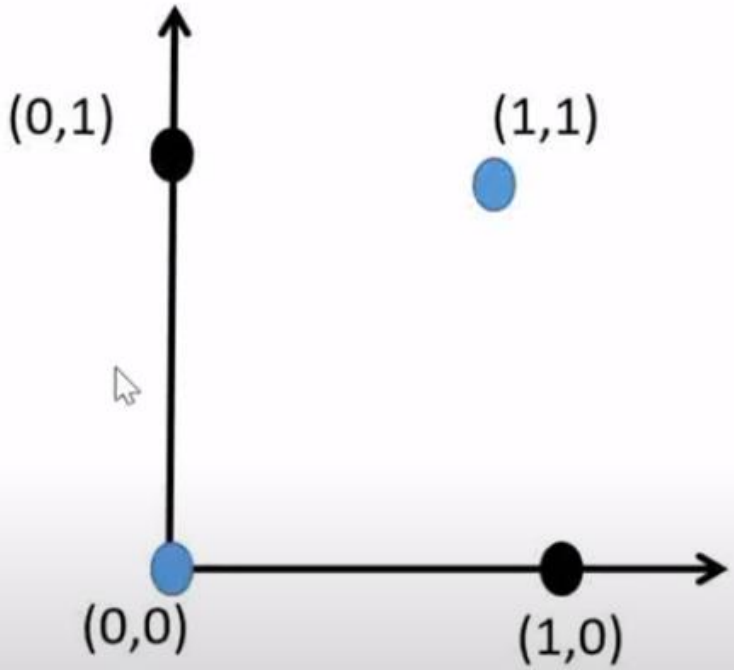
Perceptron Learning Algorithm

Proposition: If the sets P and N are finite and linearly separable, the perceptron learning algorithm updates the weight vector $\mathbf{w}_t$ a finite number of times. In other words: if the vectors in P and N are tested cyclically one after the other, a weight vector $\mathbf{w}_t$ is found after a finite number of steps t which can separate the two sets.

Non-Linearly Separable

x1	x2	y
1	1	0
1	0	1
0	1	1
0	0	0

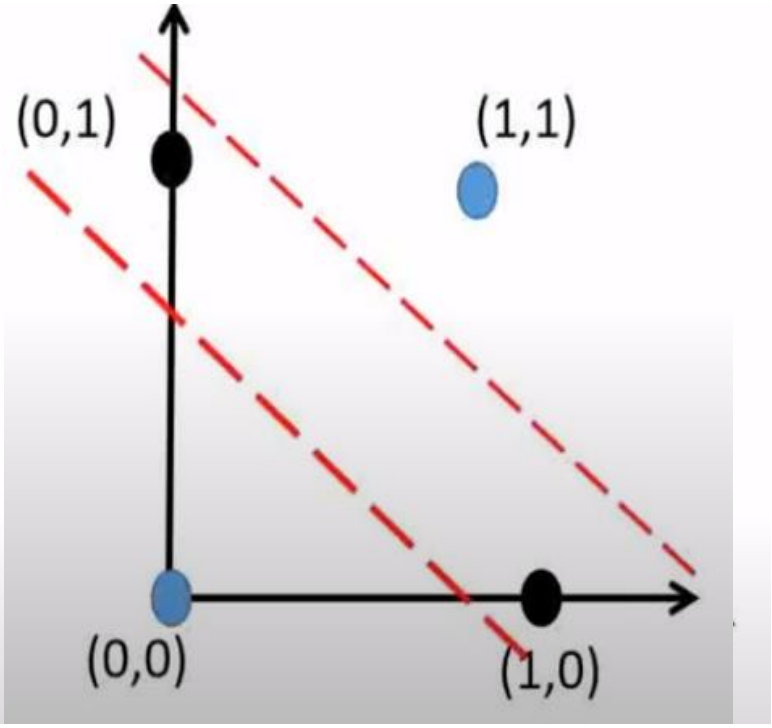
LOGIC
XOR



Non-Linearly Separable

x1	x2	y
1	1	0
1	0	1
0	1	1
0	0	0

LOGIC
XOR



Thank You!