Neuro-computing : an introduction

Susmita Ghosh
Computer Science & Engineering Department
Jadavpur University
Kolkata

- Primary task of all biological neural systems is to control various functions (mainly behavioral).
- Human being can do it almost instantaneously and without much effort. e.g., recognizing a scene or music immediately.
- **Artificial Neural Network** (ANN) or Neural Network (NN) models try to **simulate** the **biological neural network** with electronic circuitry.
- Also known as **Connectionists Model/ Parallel Distributed Processing** (PDP).

Purpose : To achieve human like performance (particularly in pattern recognition & image processing).

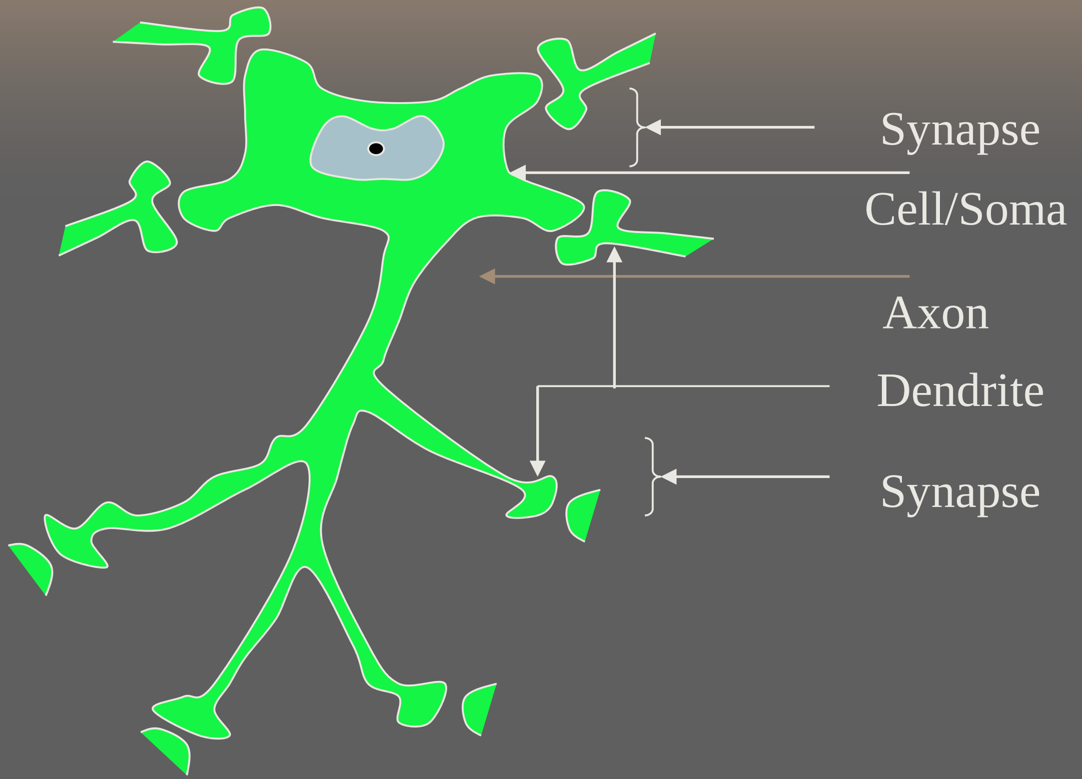
Definition

Definition : Massively parallel interconnected network of simple processing elements which are intended to interact with the objects of the real world in the same way as biological systems do.

- **NN** models are extreme simplifications of human neural systems.

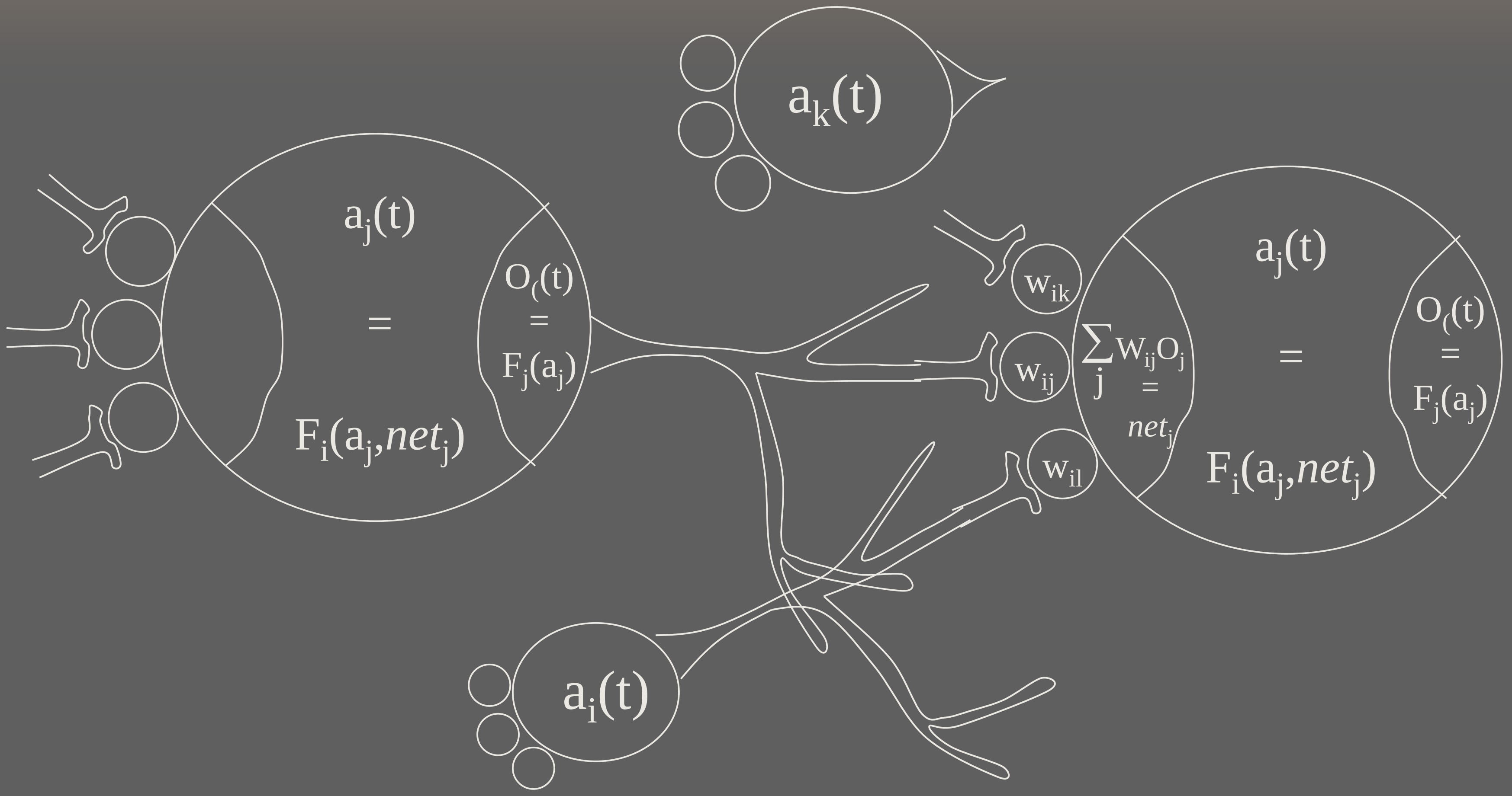
- Computational elements (neurons/nodes/ processors) are **analogous** to that of the fundamental constituents (**neurons**) of the biological nervous system.

Similarity between BNN and ANN



- Gets input via synaptic connection
- Accumulated input is transformed to a single output
- Output is transmitted through axon
- If input > 0, the neuron fires
- Total output $\Leftarrow$ firing rate

Neural network



Summary

- An electronic neuron emulates a biological neuron
- Artificial neurons are then connected to form a network to mimic (!) the topology of human nervous system
- Functions performed by a NN is determined by the network *topology*, *connection strength*, *processing* performed at computing elements or nodes, and *status updating rule*

General framework of neural networks

- **Processing units**
  - Receives input from connected neurons, compute an output value and sends it to other connected neurons.
  - Three types of units - *input*, *output*, *hidden*.
- **Output value** - $o_i(t)=f(I_i(t))$
  - Total input for i^th neuron is I_i.
  - *f* is a *threshold* or *squashing* function.
- **Unidirectional connections** (w_{ij})
  - $w_{ij} < 0 \rightarrow$ unit u_j inhibits unit u_i.
  - $w_{ij} = 0 \rightarrow$ unit u_j has no direct effect on unit u_i.
  - $w_{ij} > 0 \rightarrow$ unit u_j excites unit u_i.

Characteristics of neural networks

Exhibit a number of human brain's characteristics (partially).

- **Learn from example** - shown a set of inputs, they self-adjust to produce consistent response.
- **Generalize from previous examples to new ones** - once trained, a network's response is mostly insensitive to variations in input.
- **Abstract essential characteristics from inputs** - find the ideals (prototype) from imperfect inputs.

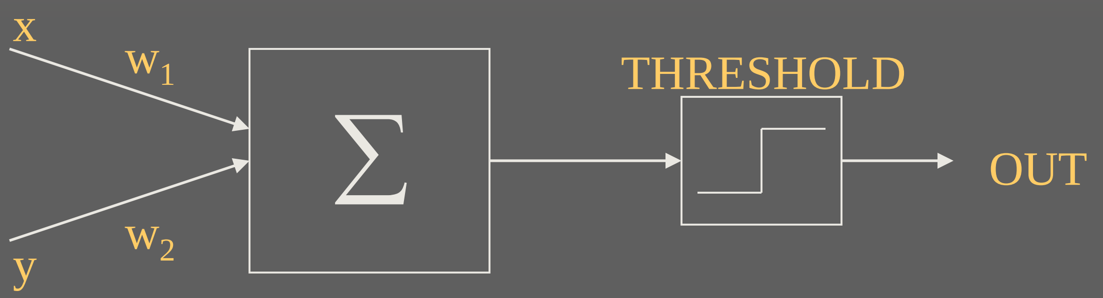
Popularly used NN models

Some common feature are there; but **differ in finer details**.

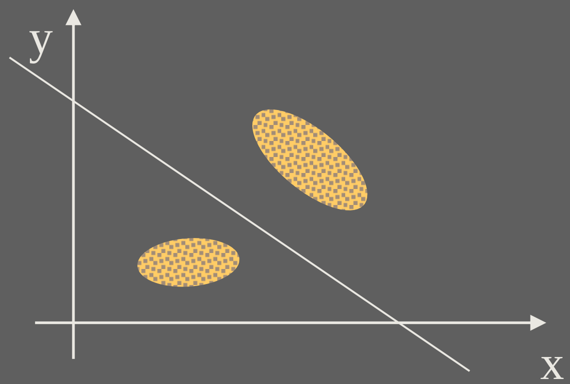
- Multi-layer perceptron (hetero associator/supervised classifier)
- Hopfield's model of associative memory (auto associator/CAM)
- Kohonen's model of self-organizing neural network (regularity detector/ unsupervised classifier)
- Radial basis function network (supervised)
- Adaptive resonance theory (regularity detector)
- Cellular neural network
- Neo-cognitron

Two input perceptron

Perceptron: A single neuron connected by weights to a set of inputs



- Let x & y be two inputs and w_1,w_2 be the weights.
- If $w_1x + w_2y > \theta$ then the output is **1** else **0**, where θ= threshold
- $w_1x + w_2y = \theta \rightarrow$ separating line



Learning rule

Learning: Present a set of input patterns, adjust the weights until the desired output occurs for each of them.

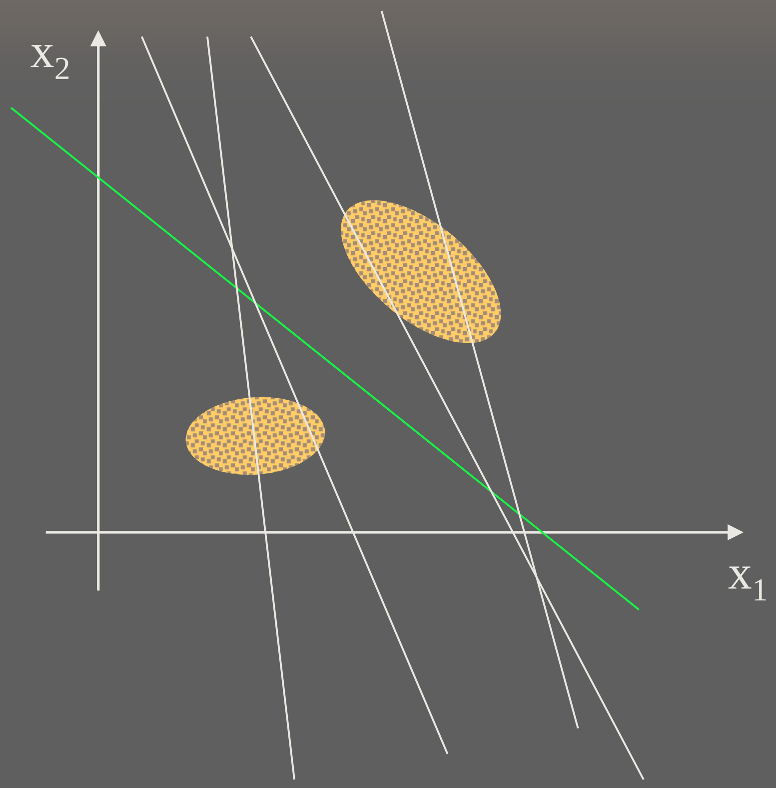
$w_i(t+1) = w_i(t) + \Delta_i$;

$\Delta_i = \eta \, \delta \, x_i$;

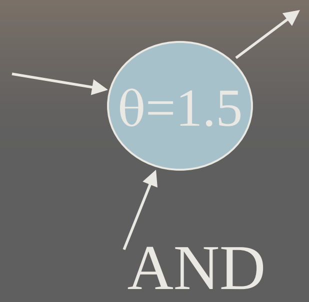
$\delta = T - A$ (i.e., target – actual).

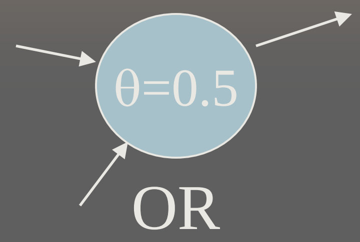
❖ If the sets of patterns are linearly separable, the single layer perceptron algorithm is guaranteed to find a separating hyperplane in a finite number of steps.

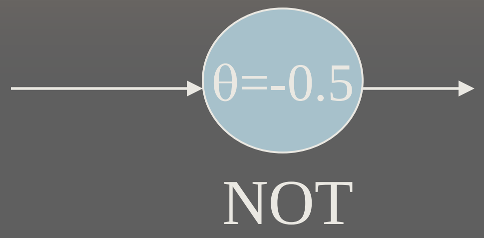
Change of weights



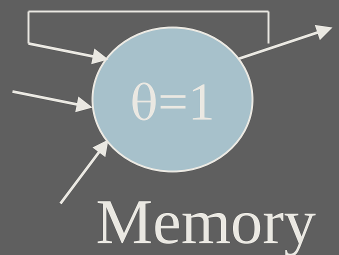
Boolean functions





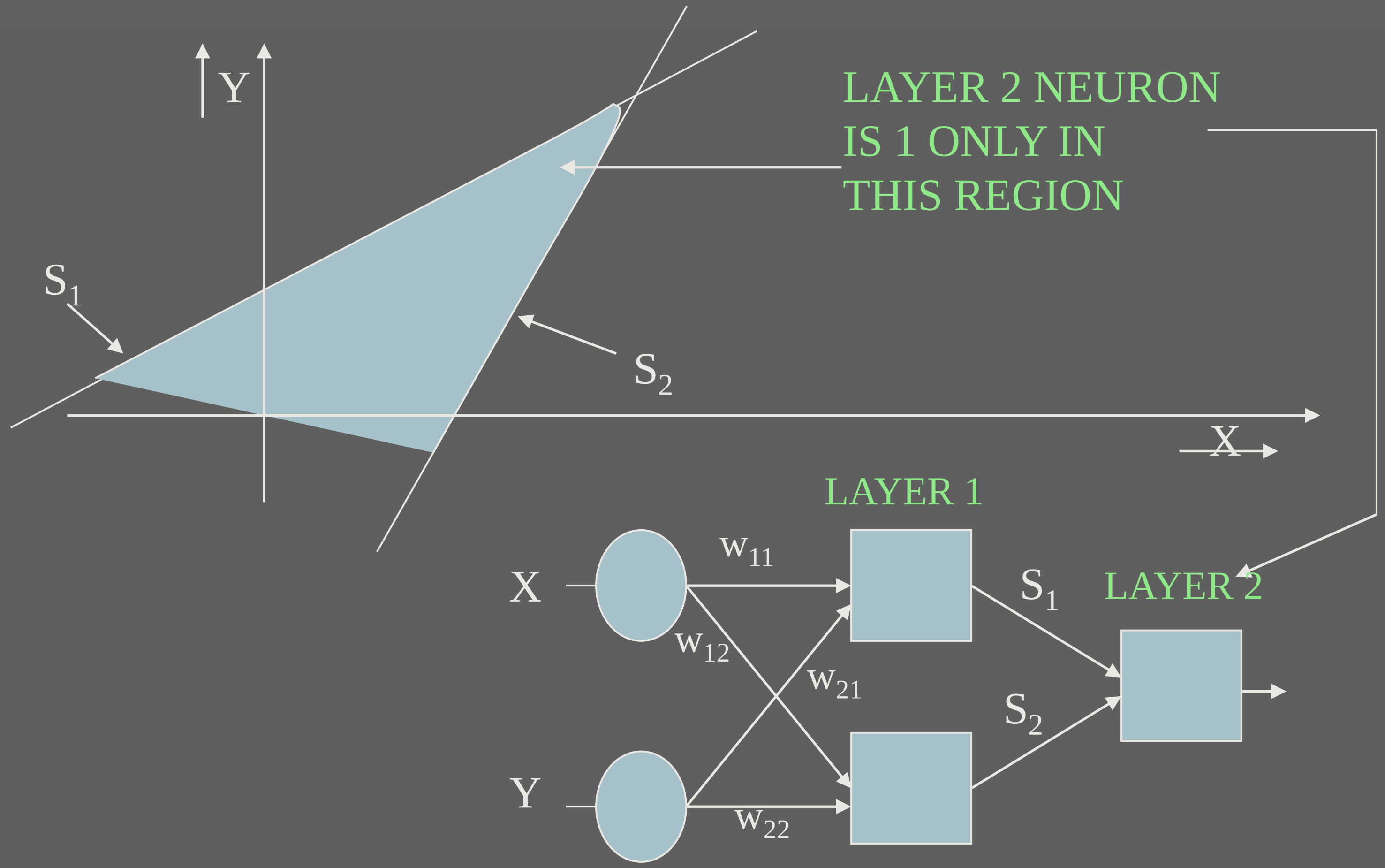


How to design other gates (NOR, NAND) ?

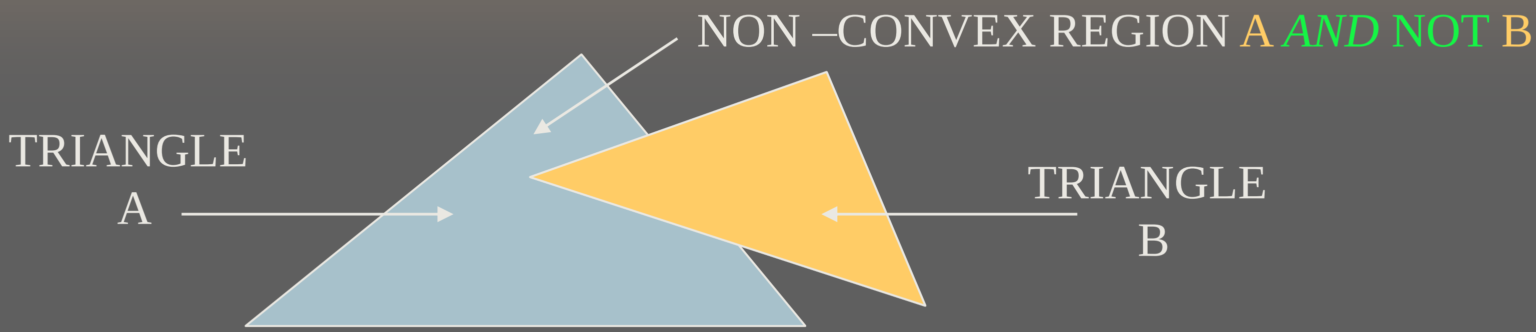


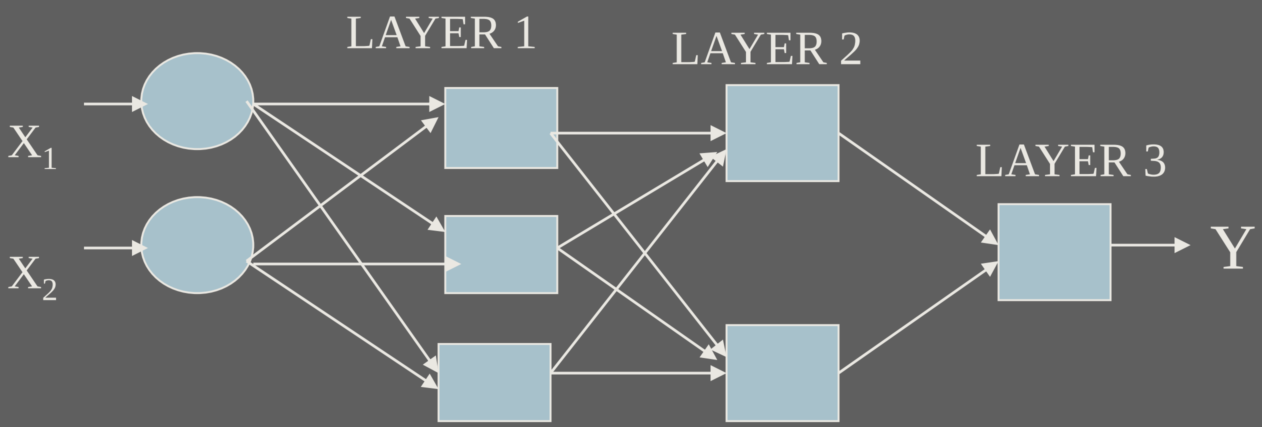
Cascading layers

Two layers : Generates convex decision regions



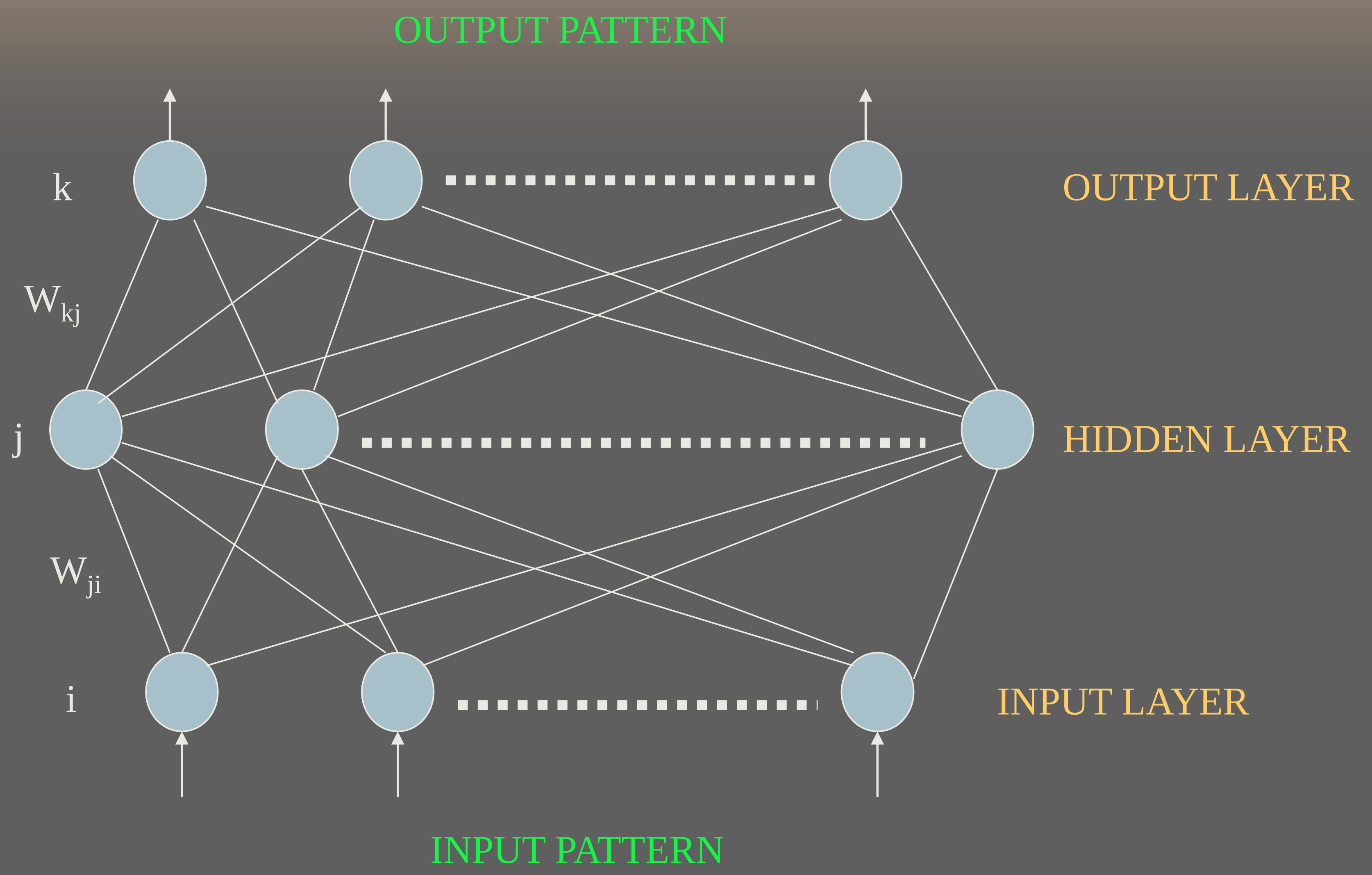
Three layers: Decision regions of any shape





Multi-layer network

Multi-layer perceptron



- Nodes of two different consecutive layers are connected by *links* or *weights*.
- There is no connection among the elements of the same layer.
- The layer where the inputs are presented is known as the *input layer*.
- On the other hand the output producing layer is called the *output layer*.
- The layers in between the input and the output layers are known as *hidden layers*.
- The total input (I_i) to the i^{th} unit

$$I_i = \sum_j w_{ij} o_j$$

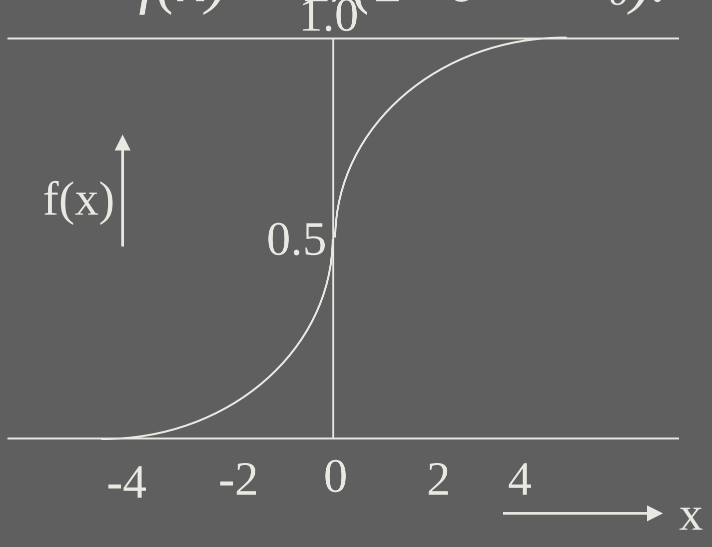
o_j is the output of the j^{th} neuron.

- The output of a node i is obtained as

$$o_i = f(I_i), \quad f \text{ is the activation function.}$$

- Mostly the activation function is sigmoidal/squashing, with the form (smooth, non-linear, differentiable & saturating),

$$f(x) = 1/(1+e^{-(x-\theta)/\theta_0}).$$



- Initially very small random values are assigned to the links/weights.

Parameter updating

- For learning (*training*) we present the input pattern $X=\{x_i\}$, and ask the net to adjust its set of weights/biases in the connecting links such that the desired output $T=\{t_i\}$ is obtained at the output layer.
- Then another pair of X and T is presented for learning.
- Learning tries to find a simple set of weights and biases that will be able to discriminate among all the input/output pairs presented to it.
- The output $\{o_i\}$ will not be the same as the target $\{t_i\}$.

  Error is,

$$E = \frac{1}{2}\sum_i (t_i - o_i)^2$$

- For learning the correct set of weights error is E is reduced as rapidly as possible.
- Use gradient descent technique.

The incremental change in the direction of negative gradient is

$$\Delta w_{ji} \propto -\frac{\partial E}{\partial w_{ji}} = -\eta \frac{\partial E}{\partial w_{ji}} = -\eta \frac{\partial E}{\partial I_j} \frac{\partial I_j}{\partial w_{ji}} = \eta \delta_j o_i$$

where $\delta_j = -\frac{\partial E}{\partial I_j} = -\frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial I_j} = -\frac{\partial E}{\partial o_j} f'(I_j).$

For the links connected to the output layer the change in weight is given by

$$\Delta w_{ji} = \eta \left(-\frac{\partial E}{\partial o_j} \right) f'(I_j) o_i .$$

For nodes in the hidden layers

$$\frac{\partial E}{\partial o_j} = \sum_k \frac{\partial E}{\partial I_k} \frac{\partial I_k}{\partial o_j} = \sum_k \frac{\partial E}{\partial I_k} \frac{\partial}{\partial o_j} \sum_i w_{ki} o_i = \sum_k \frac{\partial E}{\partial I_k} w_{kj} = \sum_k (-\delta_k) w_{kj} .$$

Hence for the hidden layer we have

$$\Delta w_{ji} = \eta \left(\sum_k \delta_k w_{kj} \right) f'(I_j) o_i$$

If $o_j = \dfrac{1}{1+e^{-\left(\sum_i w_{ji} o_i - \theta_j\right)}}$ then $f'(I_j) = \dfrac{\partial o_j}{\partial I_j} = o_j(1-o_j)$

and thus we get

$$\Delta w_{ji} = \begin{cases} \eta\left(-\dfrac{\partial E}{\partial o_j}\right) o_j(1-o_j) o_i & \rightarrow \text{output layer} \\ \eta\left(\sum_k \delta_k w_{kj}\right) o_j(1-o_j) o_i & \rightarrow \text{hidden layer} \end{cases}$$

- A large value of η corresponds to rapid learning but might result in oscillations.
- A momentum term of $\alpha \Delta w_{ji}(t)$ can be added to increase the learning rate without oscillation.

$$\Delta w_{ji}(t+1) = \eta \delta_j o_i + \alpha \Delta w_{ji}(t)$$

- The second term is used to specify that the change in w_{ji} at $(t+1)^{th}$ instant should be somewhat similar to the change undertaken at instant t.