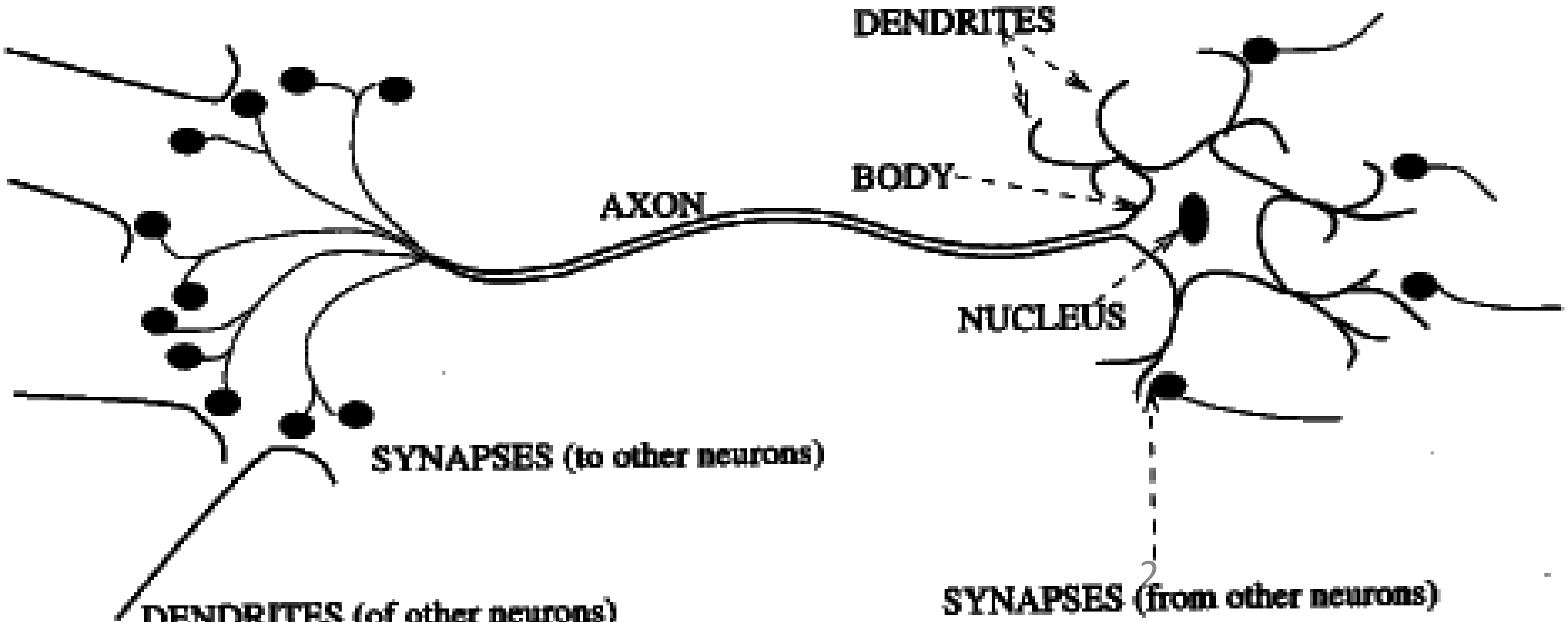
Artificial Neural Networks

- What?
  - Computing Systems inspired by Biological Neural Networks.

Biological Neural Networks

- Nervous System
  - Biological Neural Networks
    - Biological Neurons
      - What?
        - Biological Neuron is an electrically excitable cell that processes and transmits information through electrical and chemical signals.
      -



Biological Neural Networks

- Nervous System
  - Biological Neural Networks
    - Biological Neurons
      - 10 – 100 billion Neurons
      - connection to 100 – 10000 other neurons
      - 100 different types
      - layered structure

Biological Neural Networks

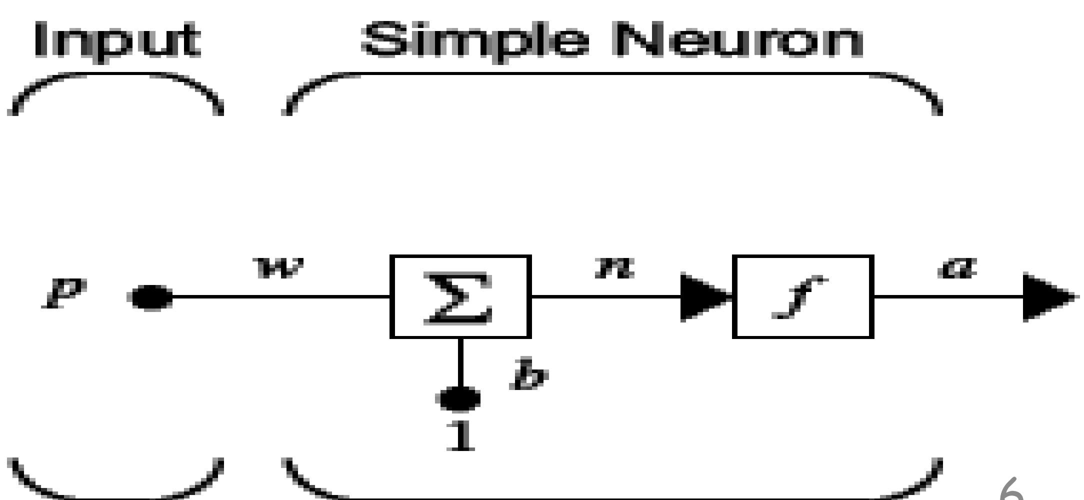
- Features
  - Parallel processing systems
  - Neurons are processing elements and each neuron performs some simple calculations
  - Neurons are networked
  - Each connection conveys a signal from one node (neuron) to another
  - Connection strength decides the extent to which a signal is amplified or diminished by a connection

Biological Neural Networks

- Features (from our experience)
  - Ability to learn from experience and accomplish complex task without being programmed explicitly
    - Driving
    - Speaking using a particular language
    - Translation
    - Speaker Recognition
    - Face Recognition, etc…

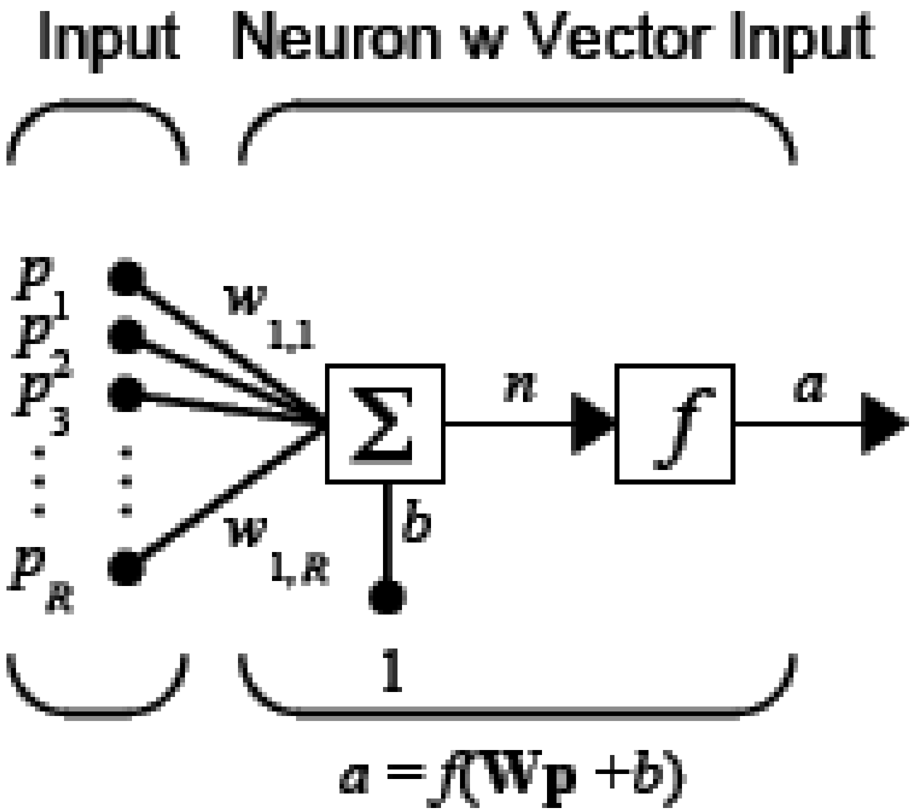
Artificial Neuron Model

- An artificial neuron is a mathematical function regarded as a model of a biological neuron.
  - Remember: 1. BN is able to receive the amplified or diminished inputs from multiple dendrites 2. It is able to combine these inputs 3. It is able to process input and produce output
- Simple Neuron
  - •Weight Function, Net Input Function & Transfer Function



$$a = f(wp + b)$$

Neuron with Vector Input

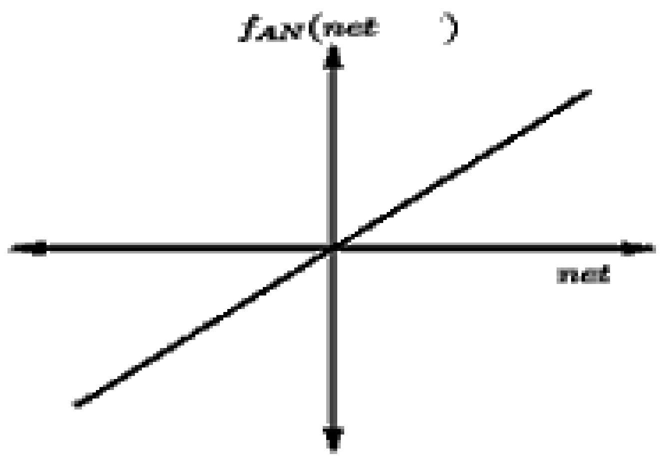


Where

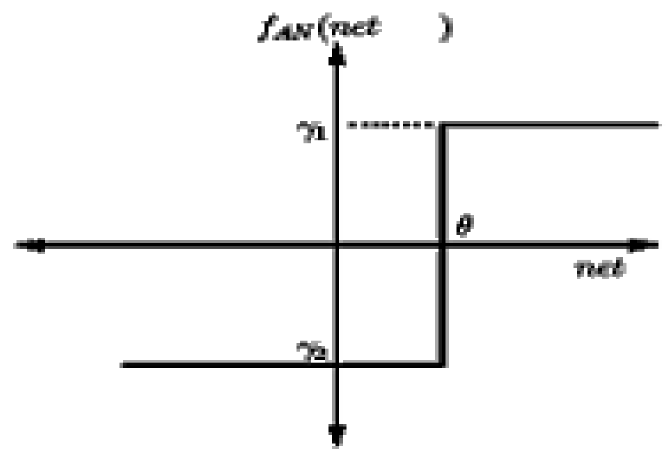
R = number of elements in input vector

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \dots + w_{1,R}p_R + b$$

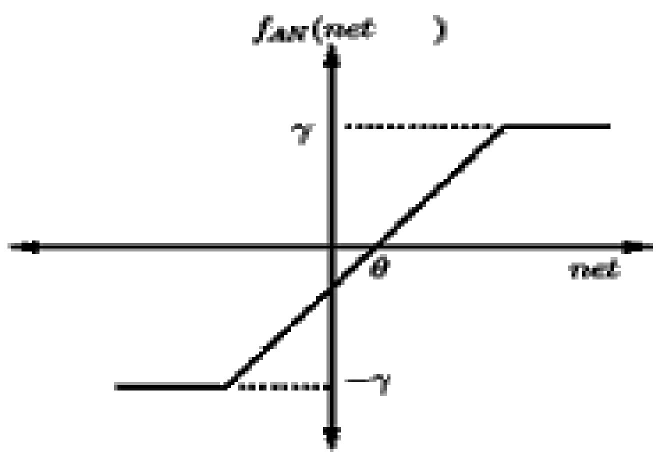
Activation Functions



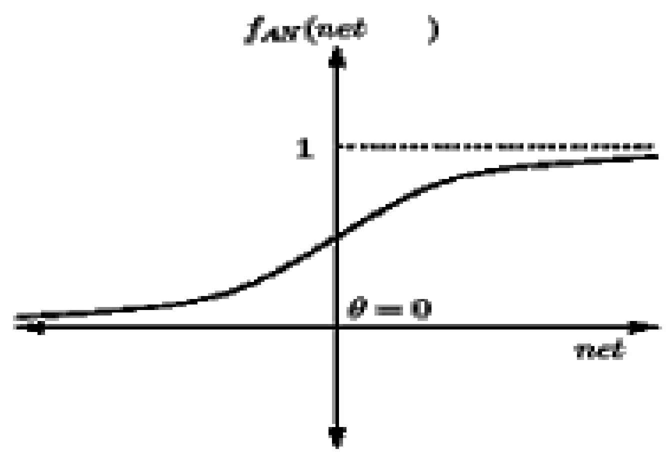
Linear function



Step function

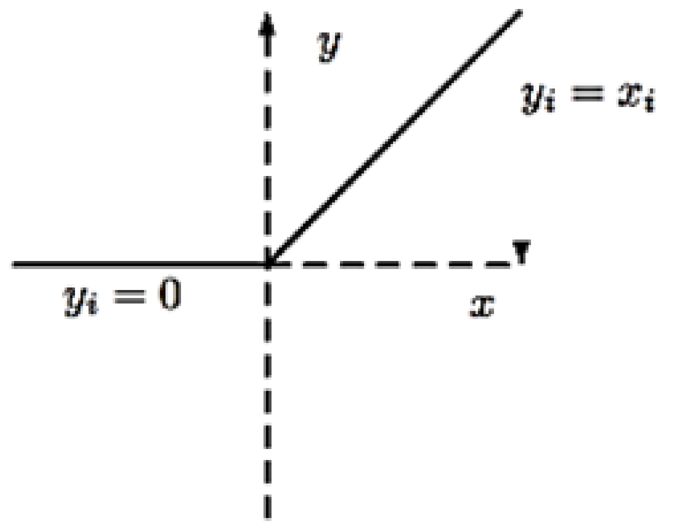


Ramp function



Sigmoid function

Activation Functions

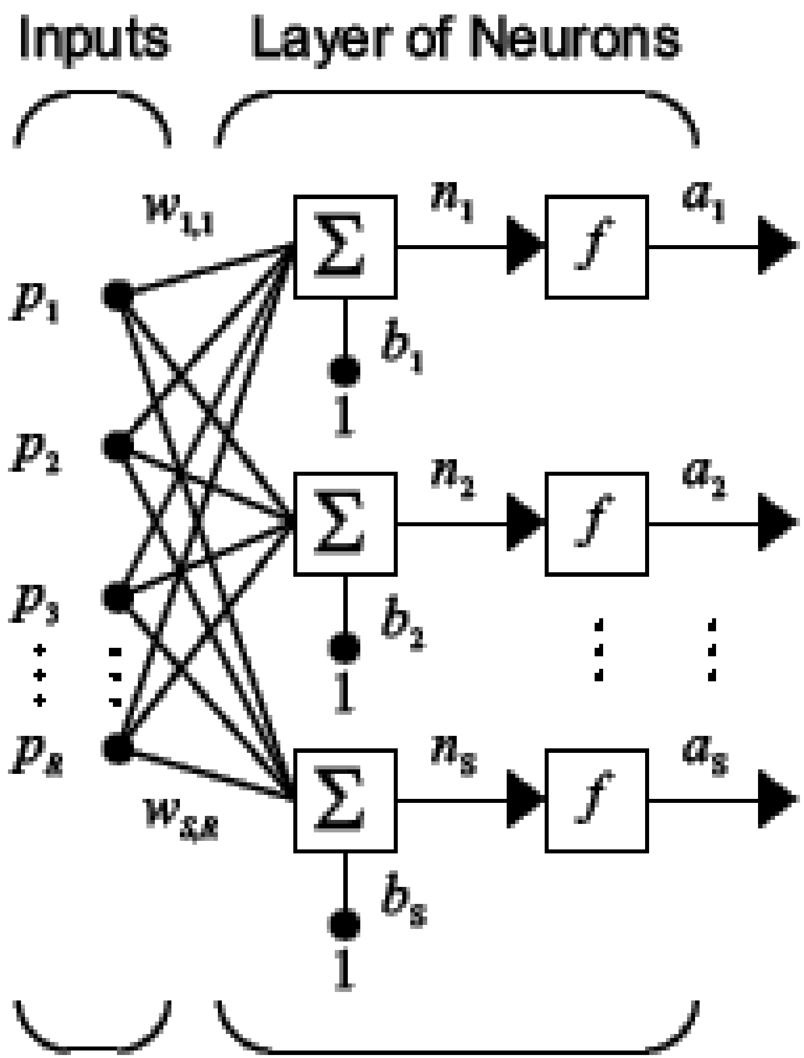


ReLU

$$y_i = \begin{cases} x_i & \text{if } x_i \geq 0 \\ 0 & \text{if } x_i < 0. \end{cases}$$

Note: Image is not Original

A Layer of Neurons



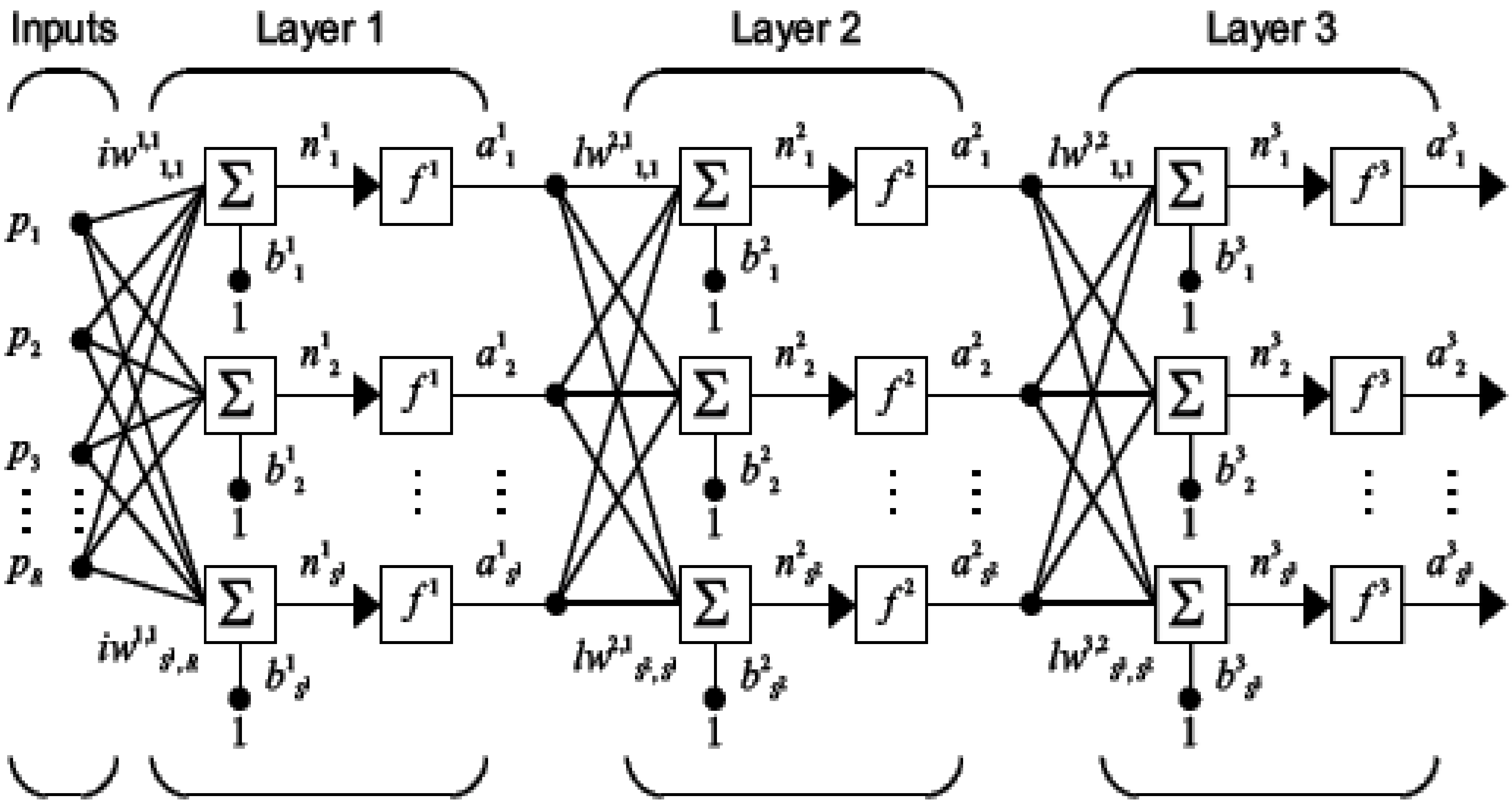
$$\mathbf{a} = \mathbf{f}(\mathbf{Wp} + \mathbf{b})$$

Where

R = number of elements in input vector

S = number of neurons in layer

Multiple Layers of Neurons

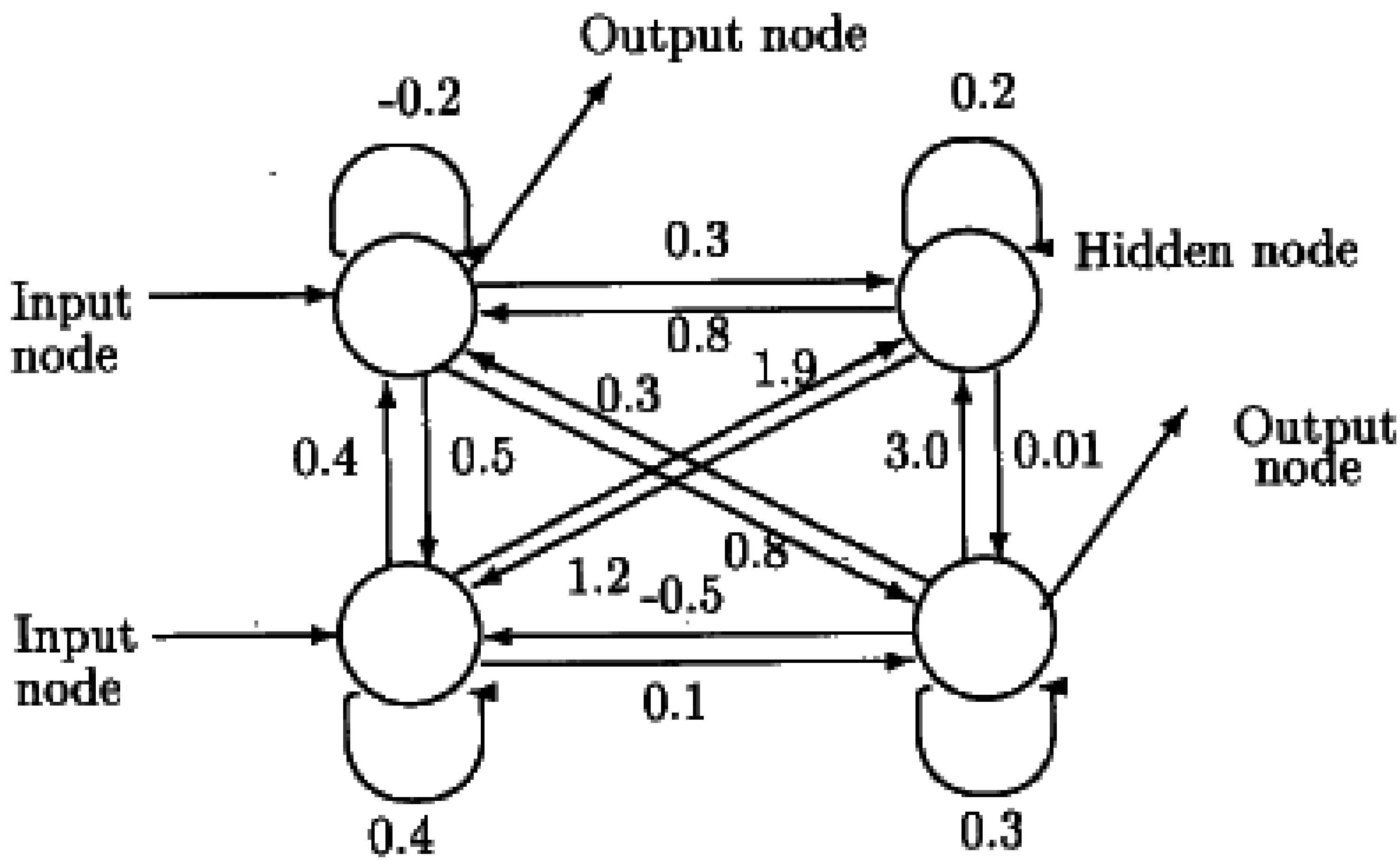


$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p}+\mathbf{b}^1) \qquad \mathbf{a}^2 = \mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{a}^1+\mathbf{b}^2) \qquad \mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{a}^2+\mathbf{b}^3)$$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

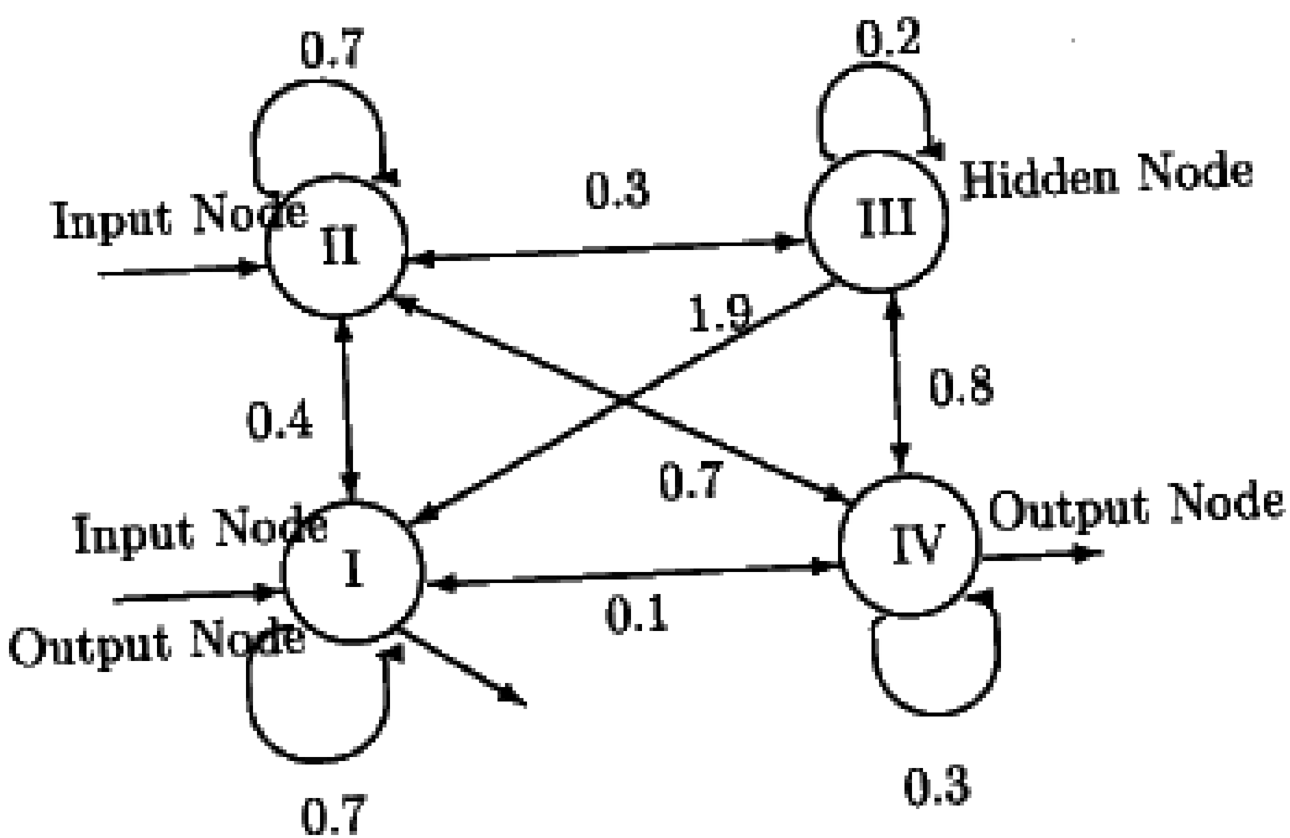
ANN Architectures

- Fully Connected Network (Asymmetric)



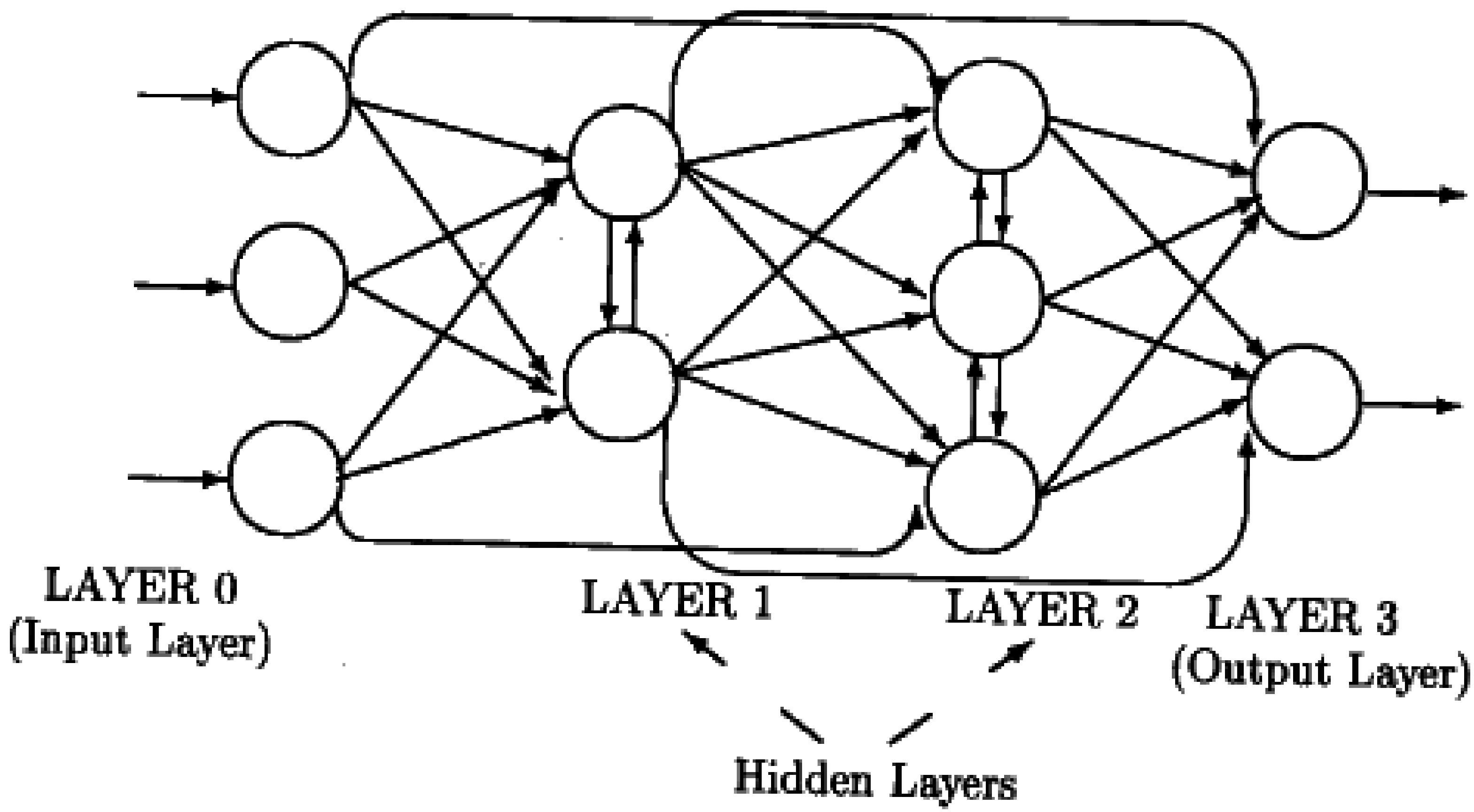
ANN Architectures

- Fully Connected Network (Symmetric)



ANN Architectures

- Layered Network



ANN Architectures

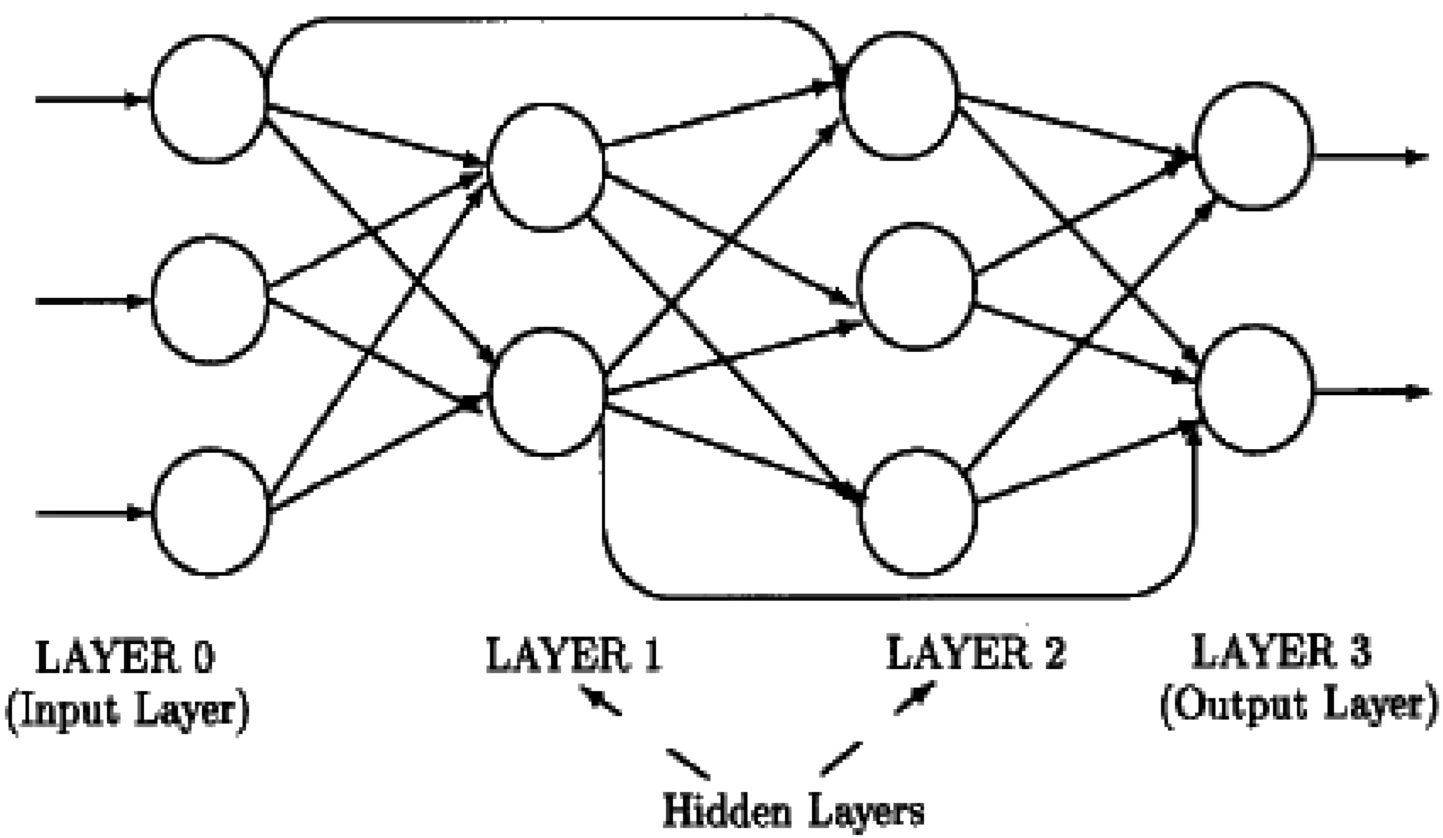
- Layered Network

13.2 Layered networks

These are networks in which nodes are partitioned into subsets called layers, with no connections that lead from layer j to layer k if $j > k$, as shown in figure 1.13.

ANN Architectures

- Acyclic Network

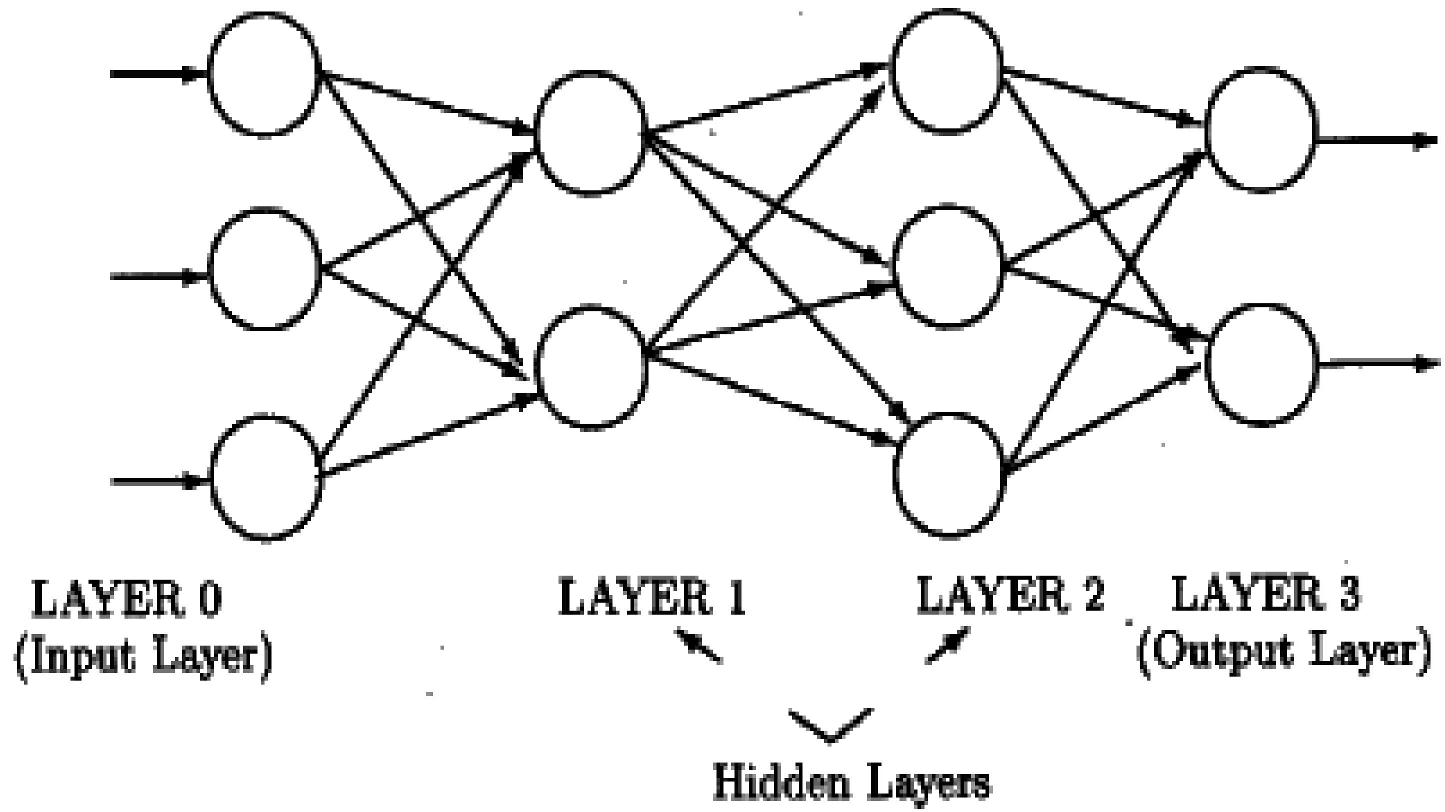


1.3.3 Acyclic networks

There is a subclass of layered networks in which there are no intra-layer connections, as shown in figure 1.14. In other words, a connection may exist between any node in layer i and any node in layer j for $i < j$, but a connection is not allowed for $i = j$.

ANN Architectures

- Feedforward Network

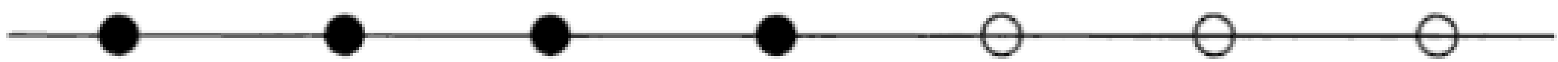


Learning in ANN

- Types of Learning
    - Supervised Learning
    - Unsupervised Learning

Linear Separability

- 1 – D Case
  - 7/5 Students data – Weight Values & Obese/Not Obese
    - (50, NO), (55, NO), (60, NO), (65, NO), (70, O), (75, O), (80, O) – Linearly Separable
    - (55, NO), (60, O), (65, NO), (70, O), (75, O) – Linearly Inseparable
  - Learning a separating point/line

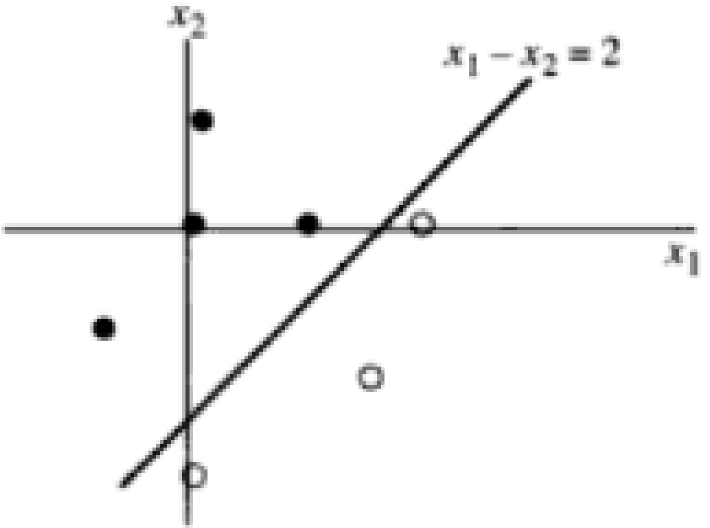


(a) Separable by a perceptron

(b) Not separable by a perceptron

Linear Separability

- 2 – D Case
  - Learning a separating line



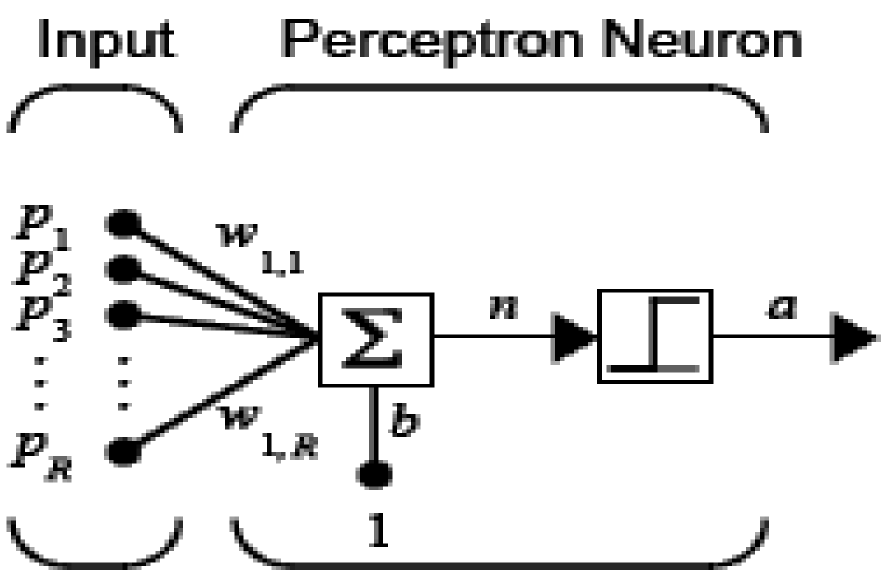
(a)

Linear Separability

- 3 – D Case
  - Learning a separating plane
- Higher Dimensional Case
  - Learning a separating hyperplane

Perceptron Model

- What is Perceptron?
  - It is a machine which can learn (using examples) to assign input vectors to different classes.
- What can it do?
  - 2-class linear classification problem
    - What?
    - Process



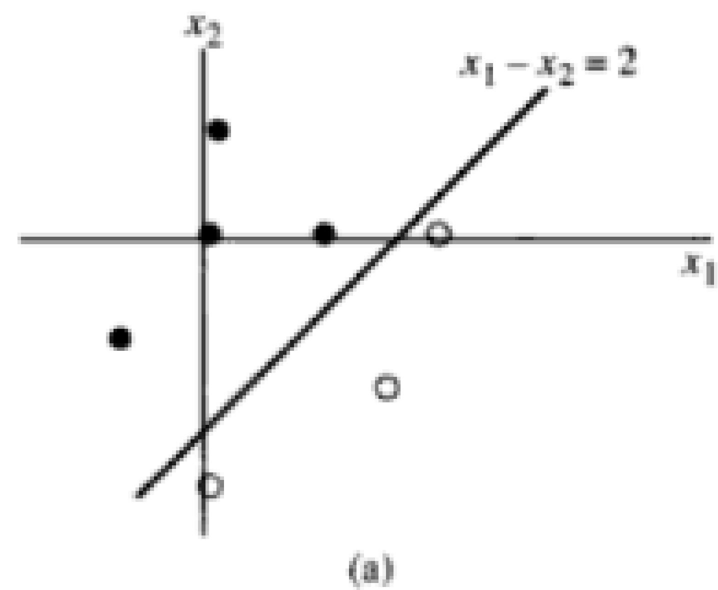
Where

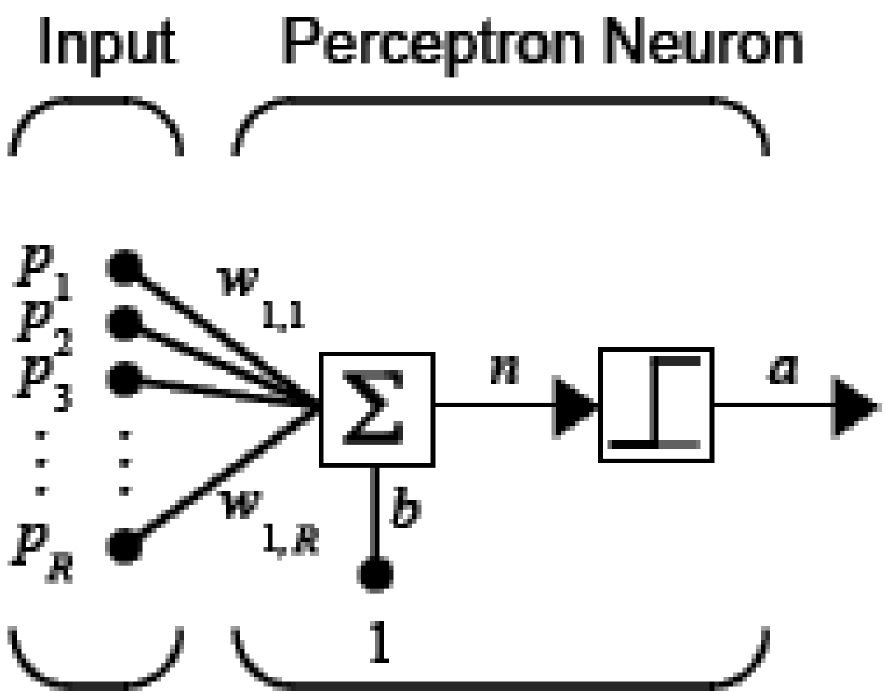
R = number of elements in input vector

$a = \mathbf{hardlim}(\mathbf{Wp} + b)$ hardlim(n) = 1, if n >= 0; 0 otherwise.

Perceptron Learning Rule

- Learning Process





R = number of elements in input vector

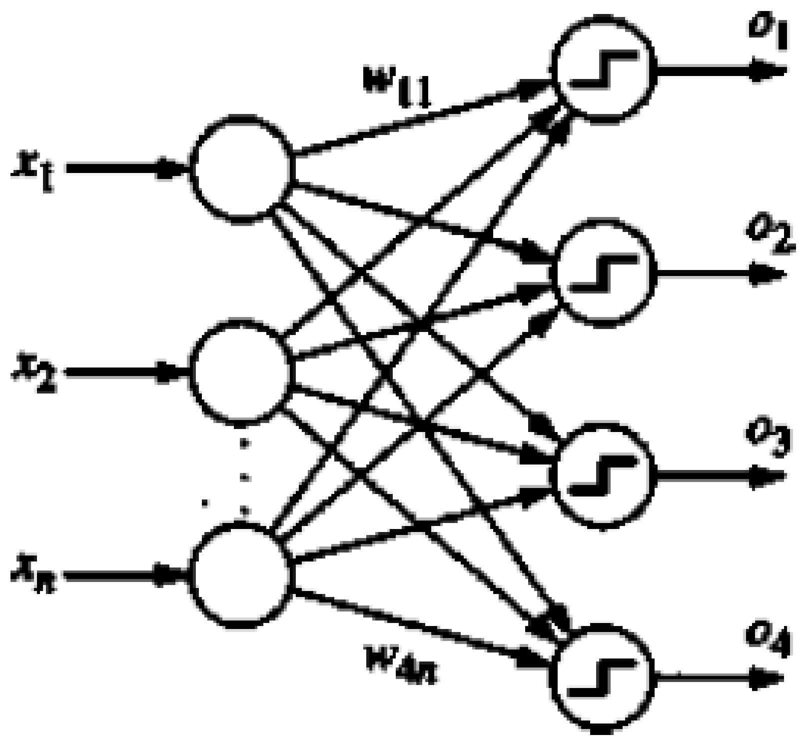
$a = \mathbf{hardlim}(\mathbf{Wp} + b)$

- $W_{new} = W_{old} + \eta eP$
- $b_{new} = b_{old} + \eta e$, where e = target - actual

Numerical

- Assume 7 one dimensional input patterns {0.0, 0.17, 0.33, 0.50, 0.67, 0.83, 1.0}. Assume that first four patterns belong to class 0 (with desired output 0) and remaining patterns belong to class 1 (with desired output 1). Design a perceptron to classify these patterns. Use perceptron learning rule. Assume learning rate = 0.1 and initial weight and bias to be (-0.36) and (-1) respectively. Show computation for two epochs.
- Order{0.83, 0.33, 0.67, 0.17, 1, 0.50, 0}

Multiclass Discrimination

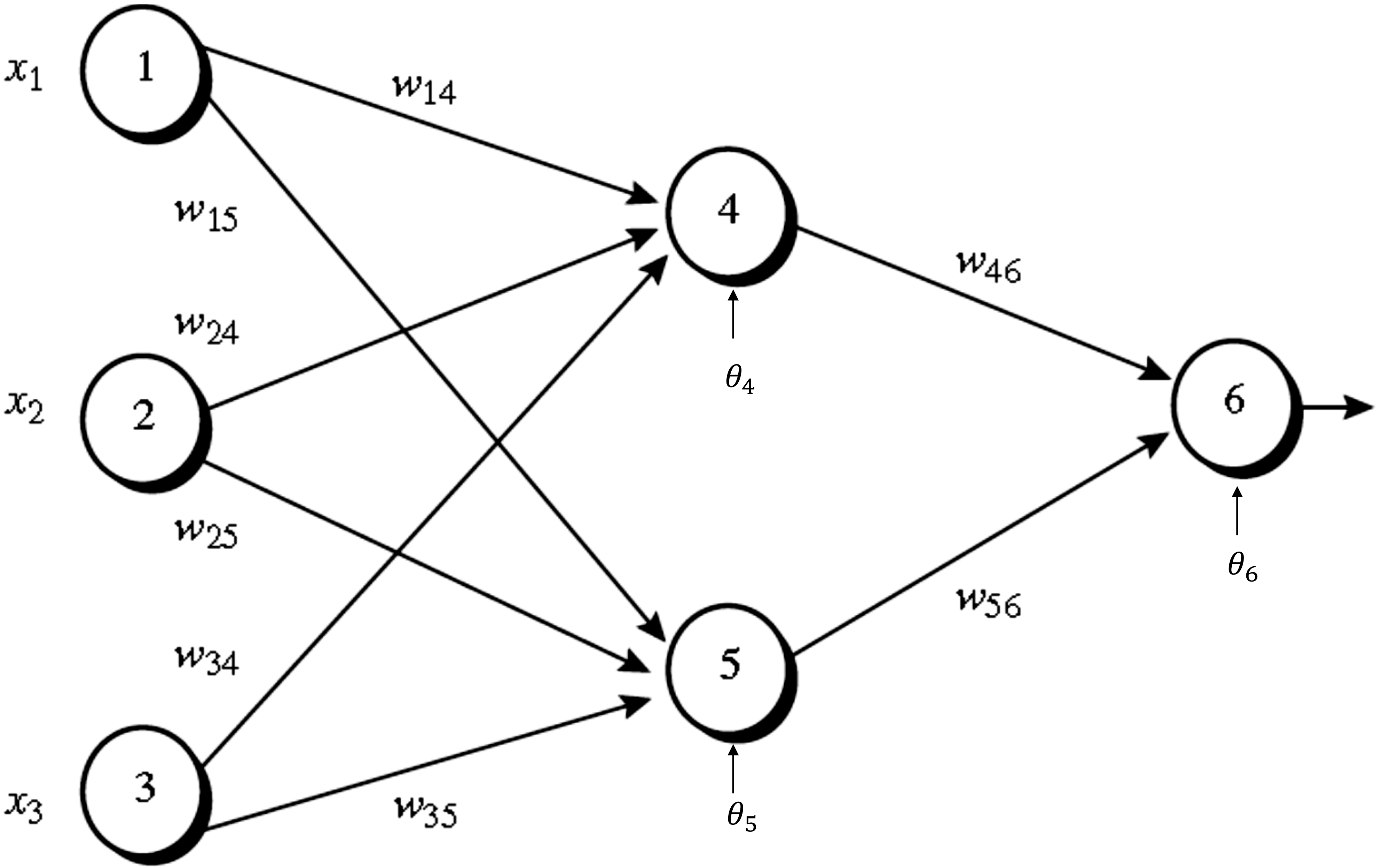
- Layer of Perceptron
  - To distinguish among n classes, a layer of n perceptrons can be used
  - 

- A presented sample is considered to belong to i^{th} calss only if i^{th} output is 1 and remaining are 0.
- If all outputs are zero, or if more than one output value equals one, the network may be considered to have failed in classification task.

Ex-OR Gate

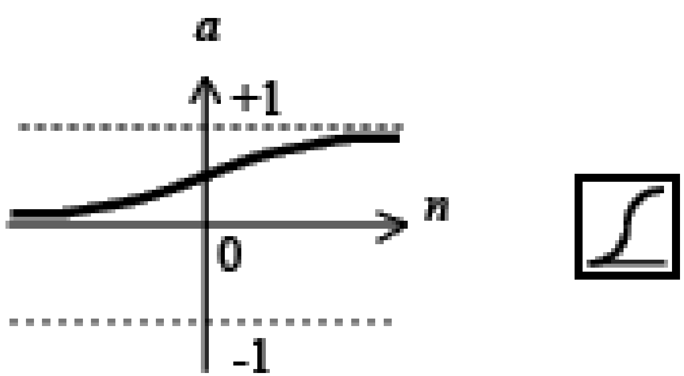
- Layer of Perceptron
  - AND Gate and OR Gate – Linearly Separable?
  - Ex-OR – Linearly Separable?
  - How to learn functionality like (classifying non-linear patterns) Ex-OR Gate?

Feed Forward Neural Network



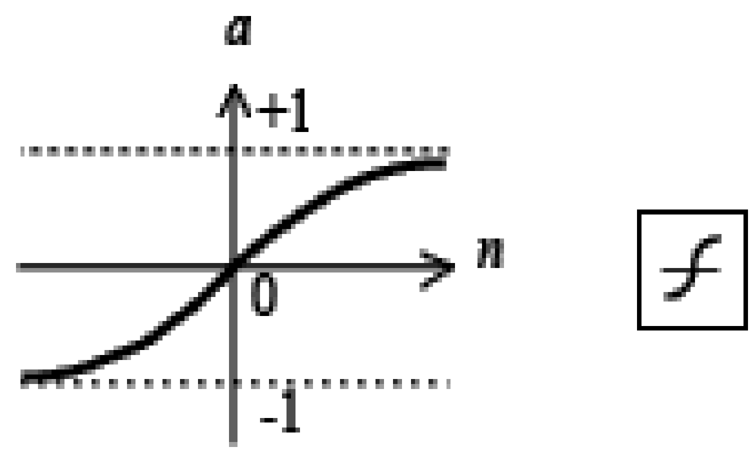
An example of a multilayer feed-forward neural network.

Multilayer Networks – Typical Transfer Functions

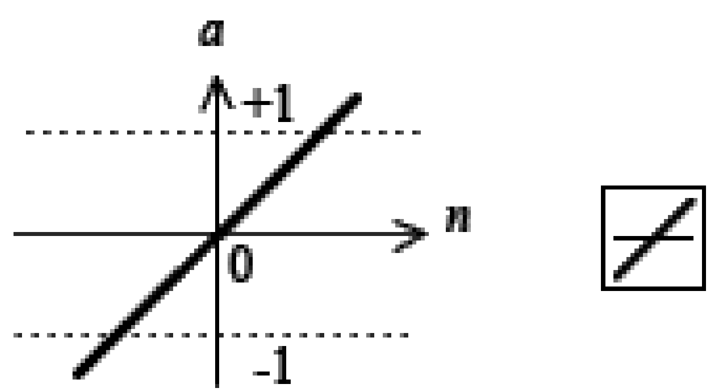


$a = logsig(n)$

Log-Sigmoid Transfer Function



$a = tansig(n)$



$a = purelin(n)$

Linear Transfer Function