Neuron Model and Network Architecture

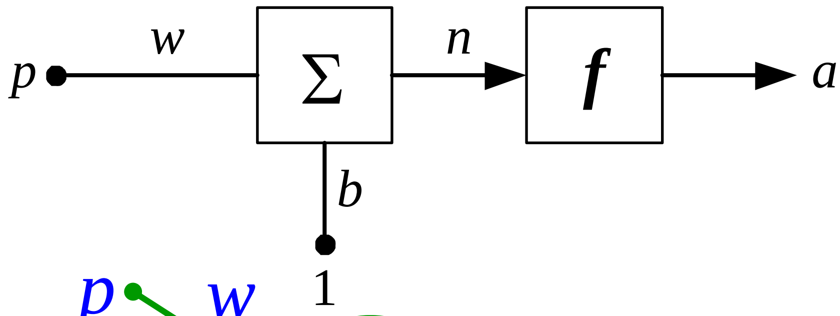
Objectives

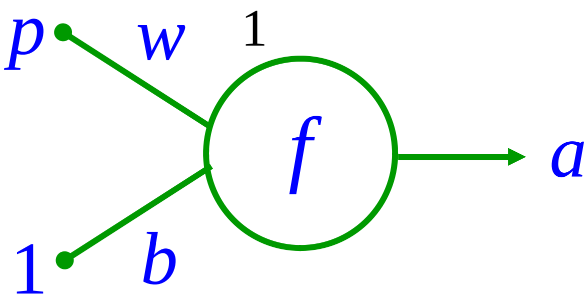
- ***Introduce*** the simplified **mathematical model** of the neuron
- ***Explain*** how these artificial neurons can be interconnected to form a variety of **network architectures**
- ***Illustrate*** the **basic operation** of these neural networks

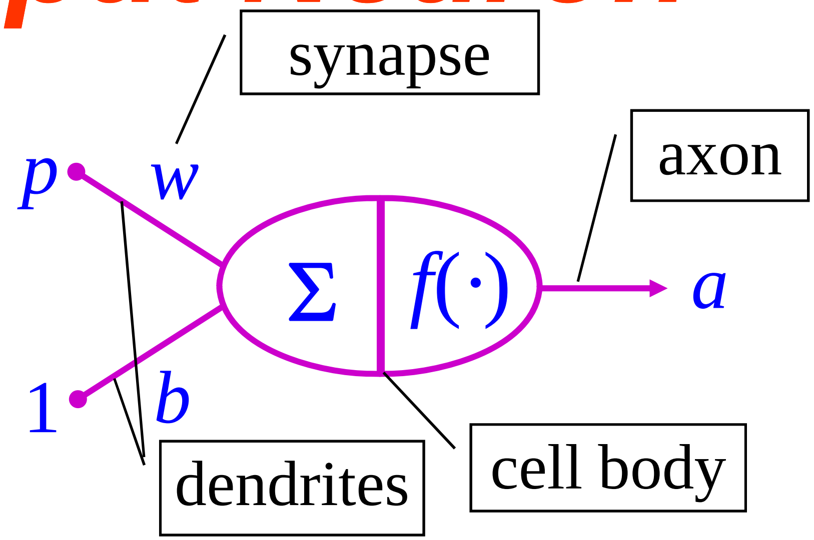
Notation

- **Scalars**: small *italic* letters
  e.g., a, b, c
- **Vectors**: small **bold** nonitalic letters
  e.g., $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$
- **Matrices**: capital **BOLD** nonitalic letters
  e.g., $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$
- Other notations are given in Appendix B

Single-Input Neuron







- $n = wp + b$
- $a = f(n) = f(wp + b)$
- $w=3$, $p=2$ and $b= -1.5$

  $\Rightarrow n = 3\times 2-1.5=4.5$

  $\Rightarrow a = f(4.5)$

- Scalar input: p
- Scalar weight: w
  (synapse)
- Bias: b
- Net input: n
- Transfer function : f
  (cell body)
- Output: a

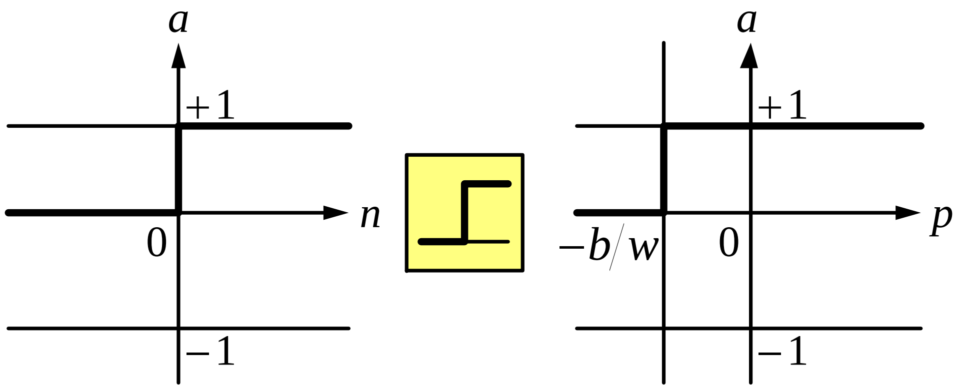
Bias and Weight

- The **bias** b is much like a **weight** w, except that it has a constant input of 1. It can be **omitted** if NOT necessary.
- **Bias** b and **weight** w are both ***adjustable*** scalar parameters of the neuron. They can be adjusted by some ***learning rule*** so that the neuron ***input/output relationship*** meets some special goal.

Transfer Functions

- The transfer function f may be a ***linear*** or ***nonlinear*** function of net input n
- Three of the most commonly used func.
  - **Hard limit** transfer function
  - **Linear limit** transfer function
  - **Log-sigmoid** transfer function

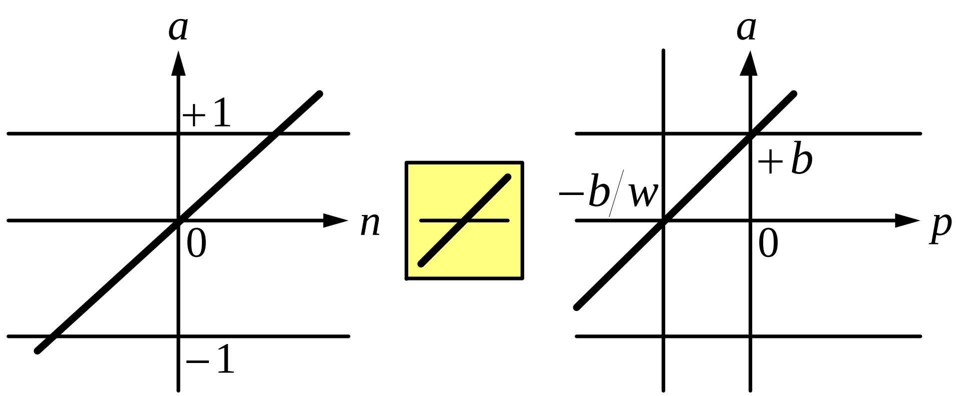
Hard Limit Transfer Func.



$a = hardlim(n)$ $a = hardlim(wp + b)$

- a = 0, if $n < 0$
- a = 1, if $n \geq 1$
- MATLAB function: *hardlim*

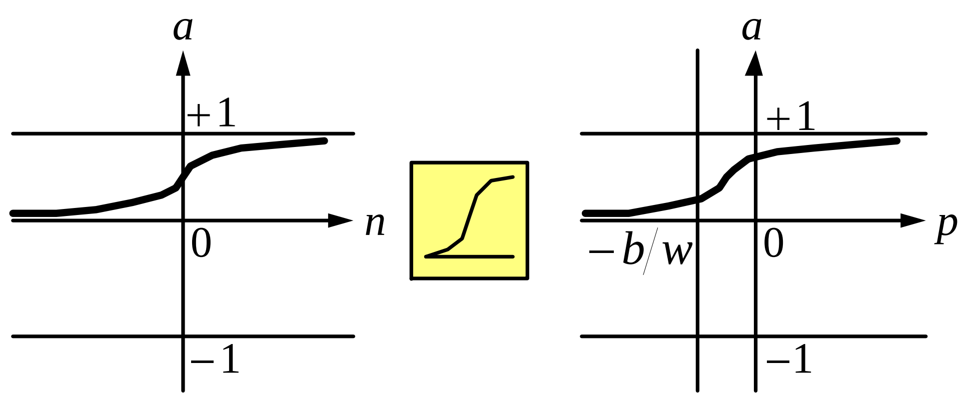
Linear Transfer Function



$a = purelin(n)$ $a = purelin(wp + b)$

- $a = n$
- MATLAB function: *purelin*

Log-Sigmoid Transfer Func.

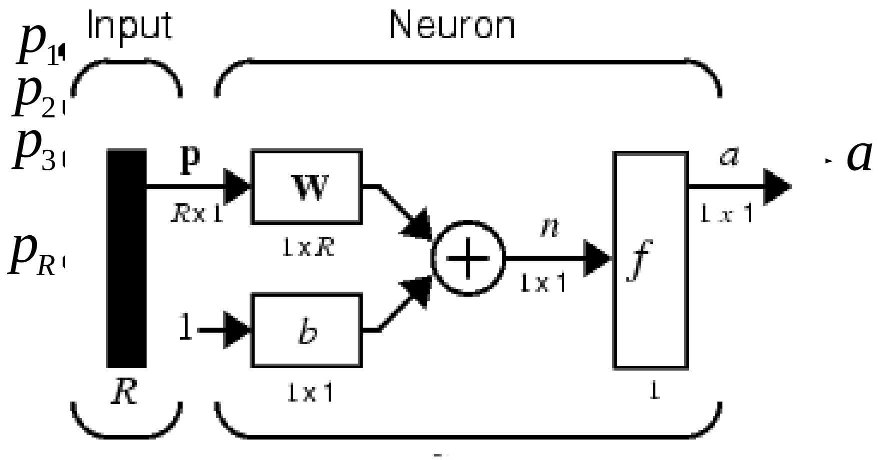


$a = logsig(n)$ $\quad$ $a = logsig(wp+b)$

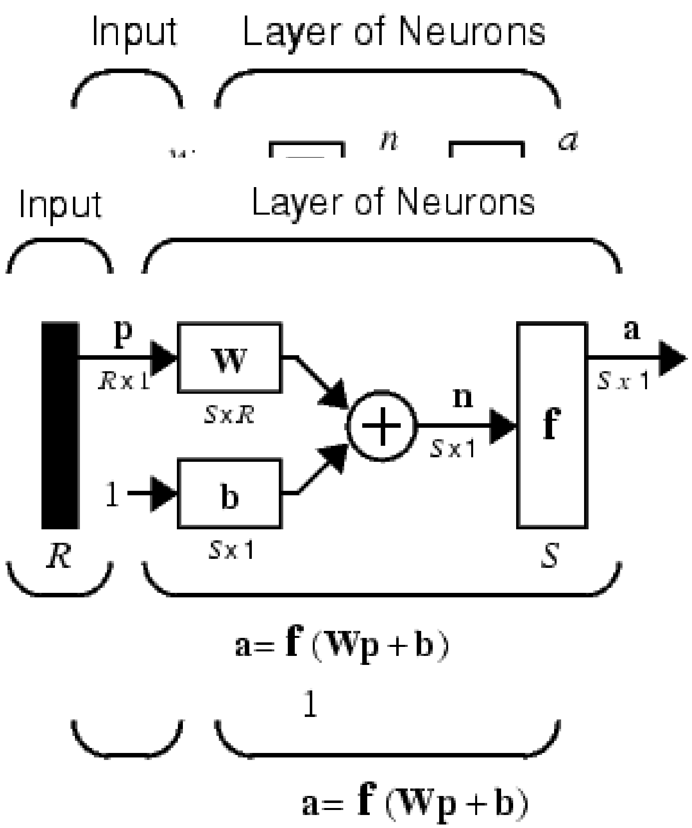
- $a = 1/[1+exp(-n)]$
- MATLAB function: *logsig*
- Other transfer functions see Pages 2-6 & 2-17

Multiple-Input Neuron

- A neuron (node) with R inputs, $p_1, p_2, \ldots, p_R$
- The weight matrix $\mathbf{W}$, $w_{11}, w_{12}, \ldots, w_{1R}$
- The neuron has a bias b
- Net input: $n = w_{11}\, p_1 + w_{12}\, p_2 + \ldots + w_{1R} p_R + b = \mathbf{Wp} + b$
- Neuron output: $a = f(\mathbf{Wp} + b)$



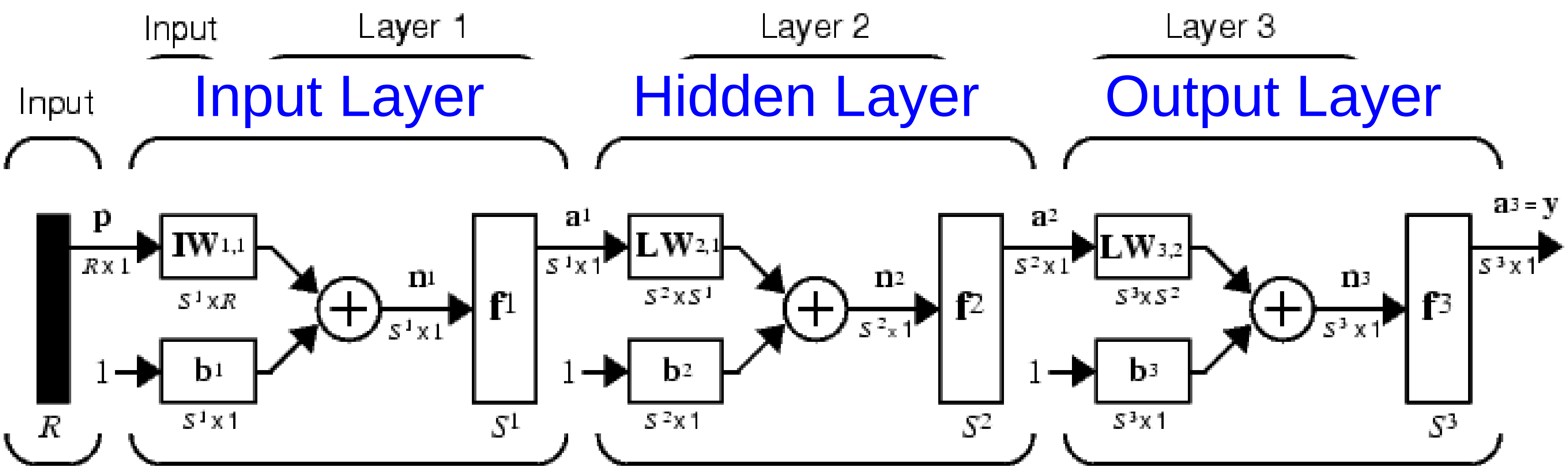
Single-Layer Network



- R: number of input
- S: number of neuron (node) in a layer ($R \neq S$)
- Input vector **p** is a vector of length R
- Bias vector **b** and output vector **a** are vectors of length S
- Weight matrix **W** is an $S \times R$ matrix

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{22} & \cdots & w_{1R} \\ w_{21} & w_{22} & \cdots & w_{2R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S1} & w_{S2} & \cdots & w_{SR} \end{bmatrix}$$

Multiple-Layer Network



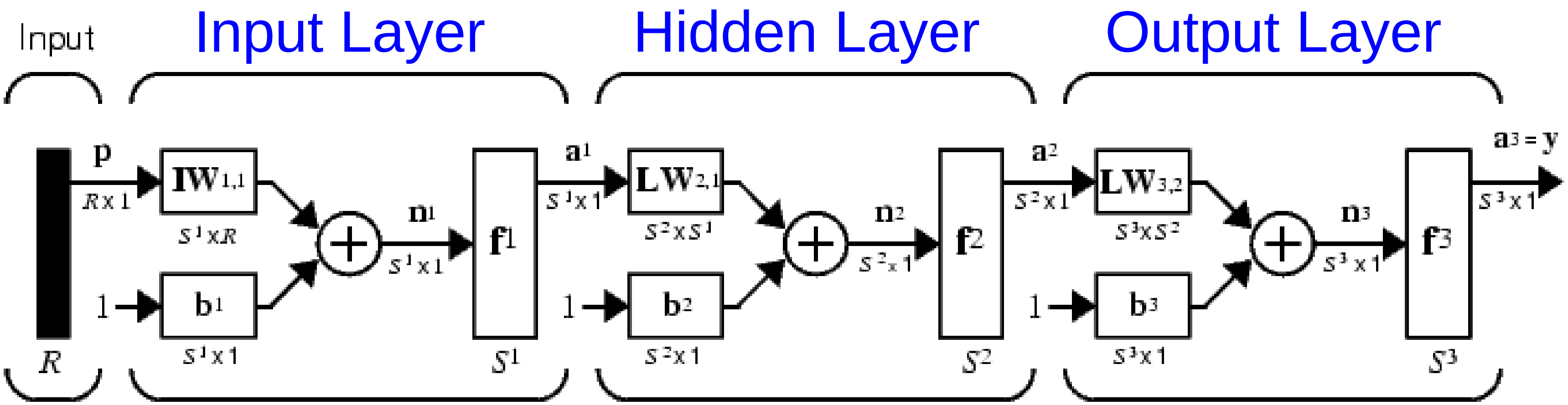
$a^1 = f^1(IW^{1,1}p + b^1)$ $a^2 = f^2(LW^{2,1}a^1 + b^2)$ $a^3 = f^3(LW^{3,2}a^2 + b^3)$

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

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

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

Multiple-Layer Network



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

- **Layer Superscript**: the number of the layer
- R inputs, S^n neurons (nodes) in the nth layer
- Different layers can have different numbers of neurons
- The **outputs** of layer k are the **inputs** of layer $(k+1)$
- **Weight matrix $\mathbf{W}^j$** between layer i and j is an $S^i \times S^j$ matrix

Network Architectures

- Models of neural networks are specified by the ***three*** basic entities: models of the ***processing elements*** (neurons), models of ***inter-connections and structures*** (network topology), and the ***learning rules*** (the ways information is stored in the network).
- The **weights** may be ***positive*** (excitatory) or ***negative*** (inhibitory).
- **Information** is stored in the ***connection weights***.

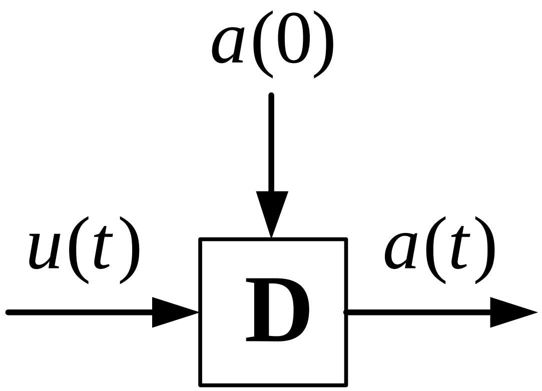
Network Structures

- The layer that receives inputs is called the ***input layer***.
- The outputs of the network are generated from the ***output layer***.
- Any layer between the input and the output layers is called a ***hidden layer***.
- There may be ***from zero to several*** **hidden layers** in a neural network.

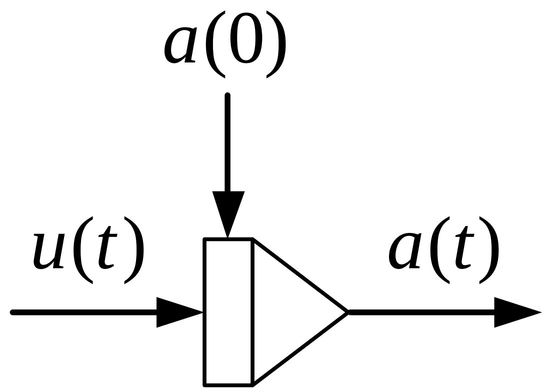
Network Structures

- When no node output is an input to a node in the same layer or preceding layer, the network is a ***feedforward network***.
- When outputs are directed back as inputs to same- or preceding-layer nodes, the network is a ***feedback network***.
- Feedback networks that have **closed loops** are called ***recurrent networks***.

Delay Block & Integrator

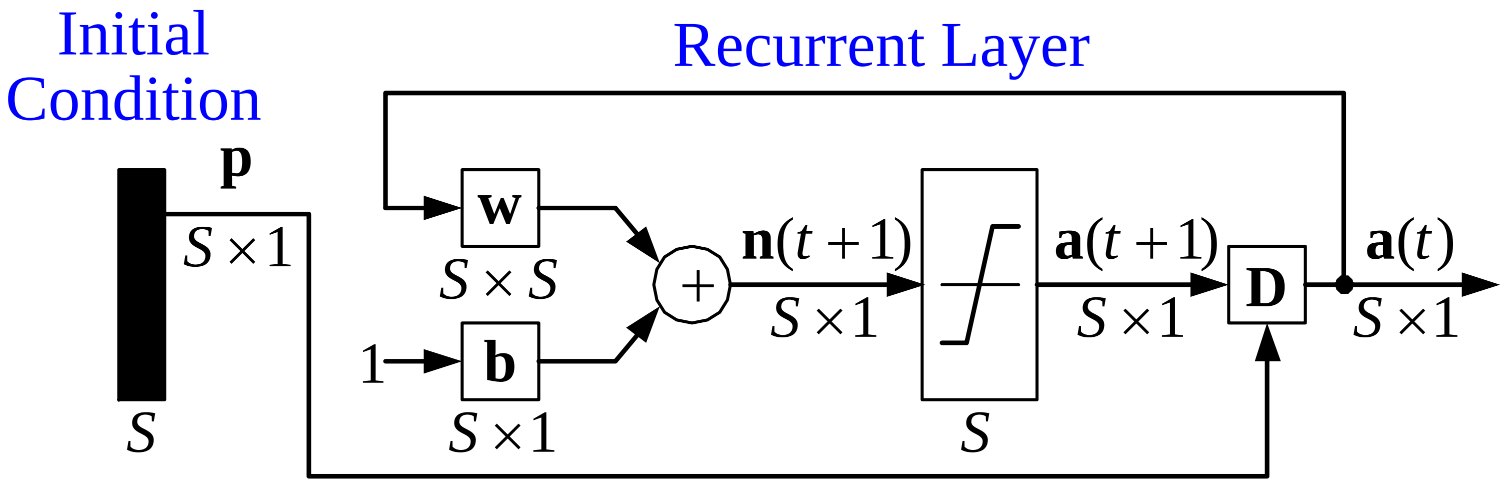


$$a(t) = u(t-1)$$



$$a(t) = \int_0^t u(\tau)\,d\tau + a(0)$$

Recurrent Network



$$\mathbf{a}(0) = \mathbf{p}$$

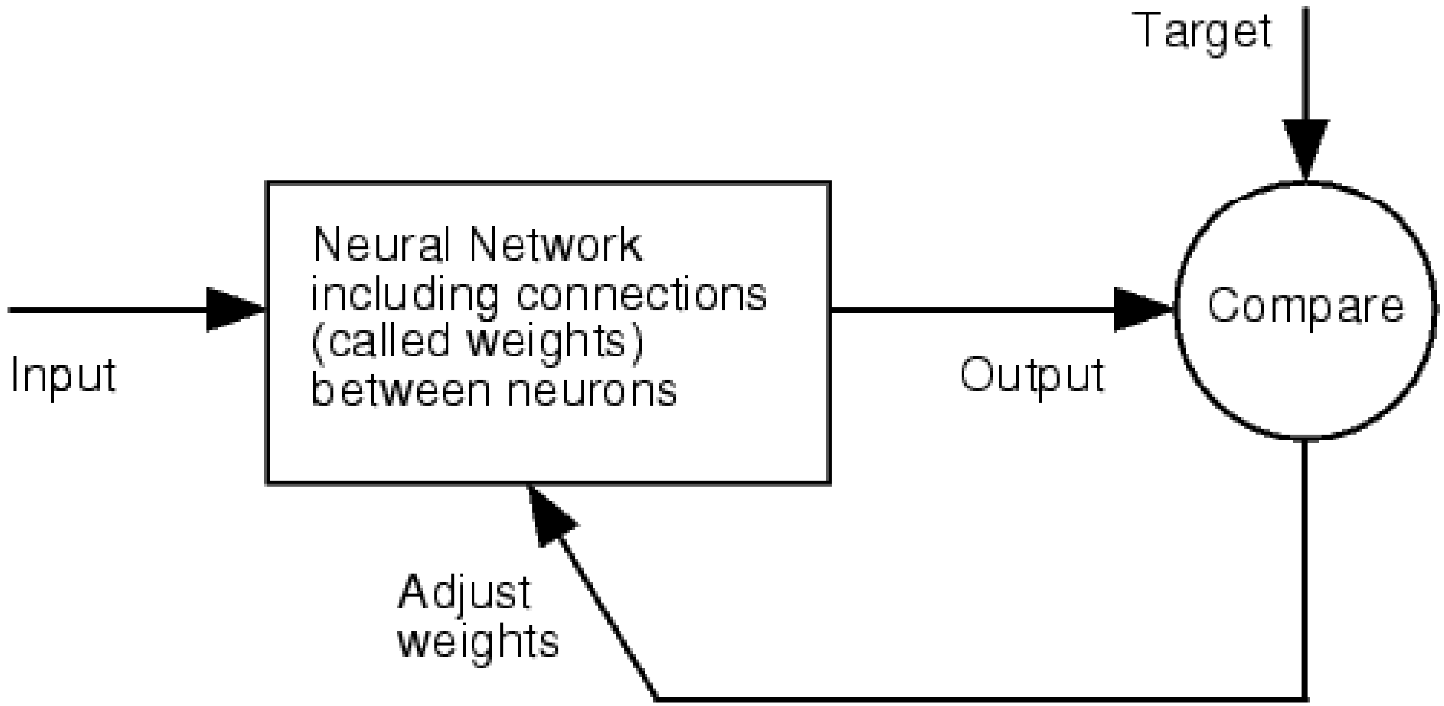
$$\mathbf{a}(1) = satlins(\mathbf{Wa}(0) + \mathbf{b}), \quad \mathbf{a}(2) = satlins(\mathbf{Wa}(1) + \mathbf{b})$$

$$\mathbf{a}(t+1) = satlins(\mathbf{Wa}(t) + \mathbf{b})$$

Learning Scheme

- **Two** kinds of learning in neural networks: ***parameter learning***, which concerns the updating the **connection weights** in a neural network, and ***structure learning***, which focuses on the change in the **network structure**, including the *number of nodes* and their *connection types*.
- Each kind of learning can be further classified into **three categories**: ***supervised learning***, ***reinforcement learning***, and ***unsupervised learning***.

Learning Scheme



How to Pick an Architecture

Problem specifications help define the network in the following ways:

1. **Number of network inputs** = number of problem inputs
2. **Number of neurons in output layer** = number of problem outputs
3. **Output layer transfer function** choice at least partly determined by problem specification of the outputs.