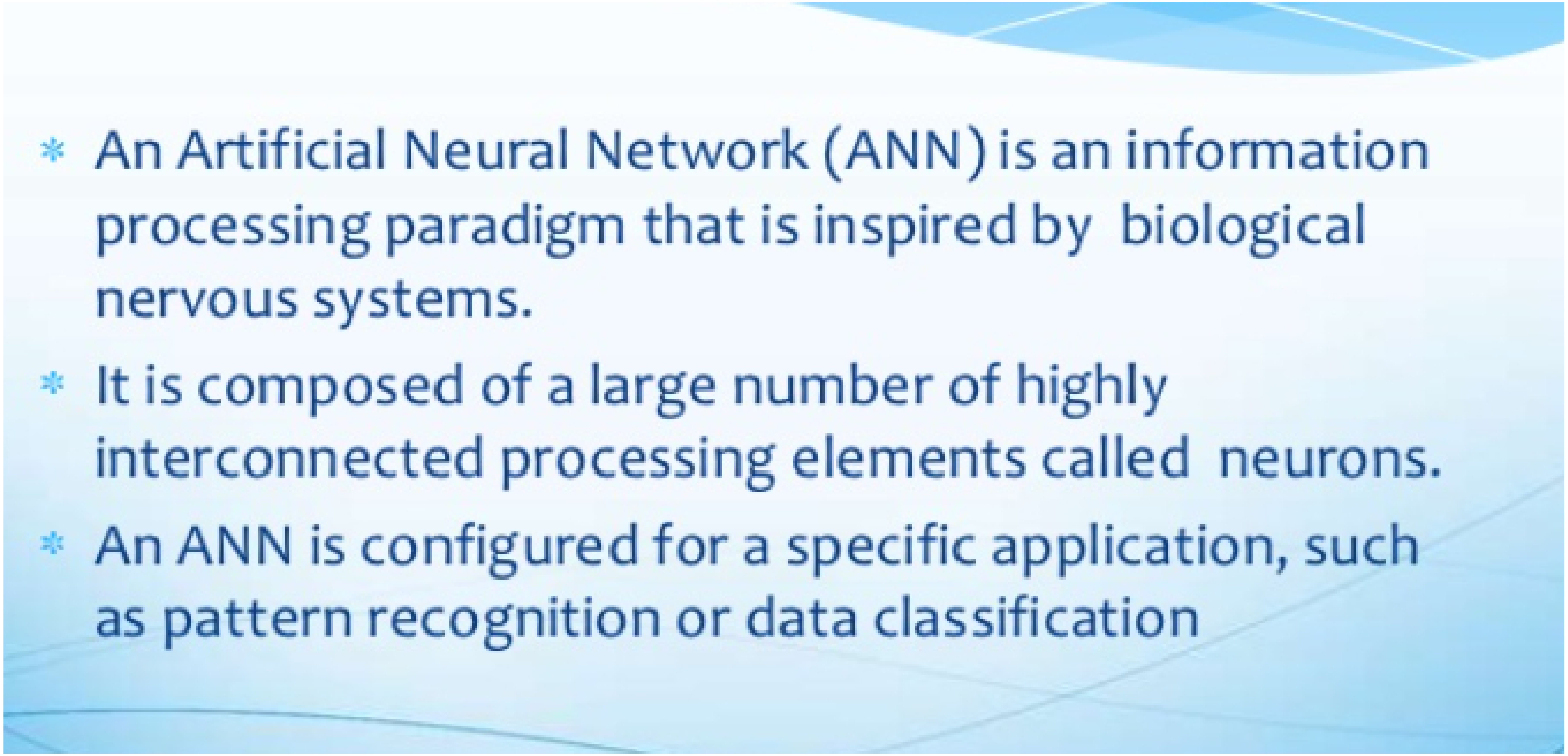
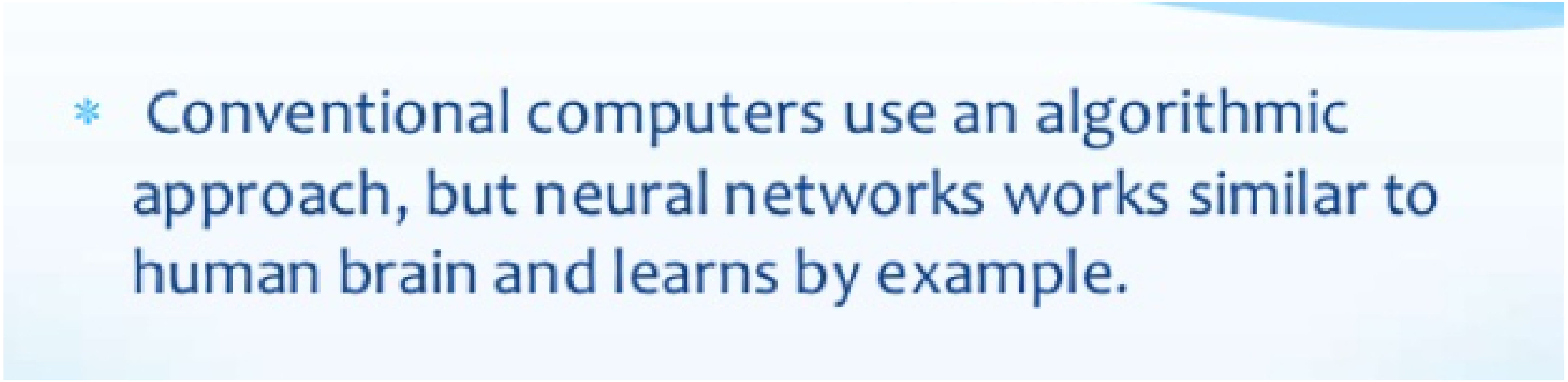
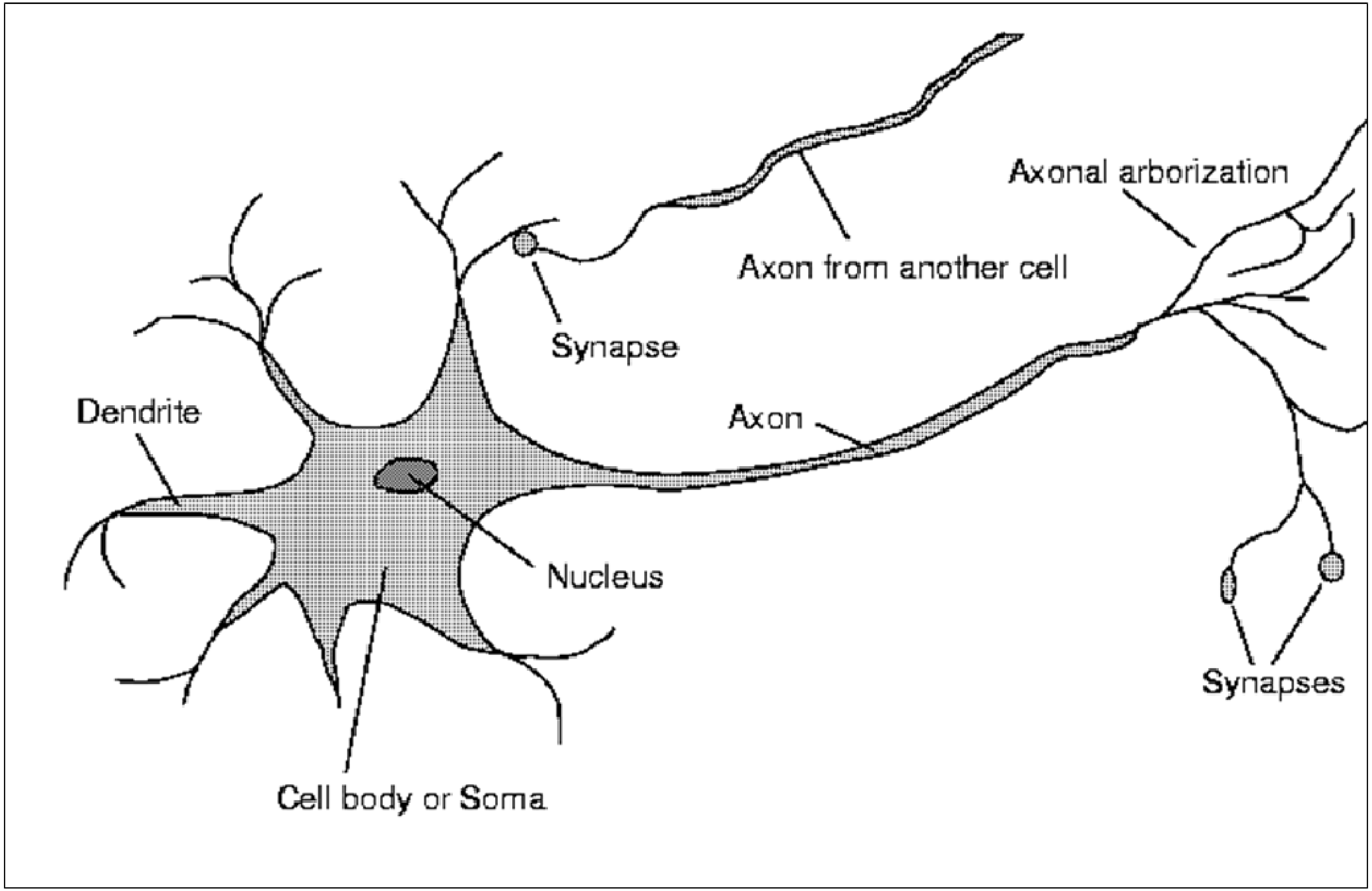


- An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by biological nervous systems.
- It is composed of a large number of highly interconnected processing elements called neurons.
- An ANN is configured for a specific application, such as pattern recognition or data classification

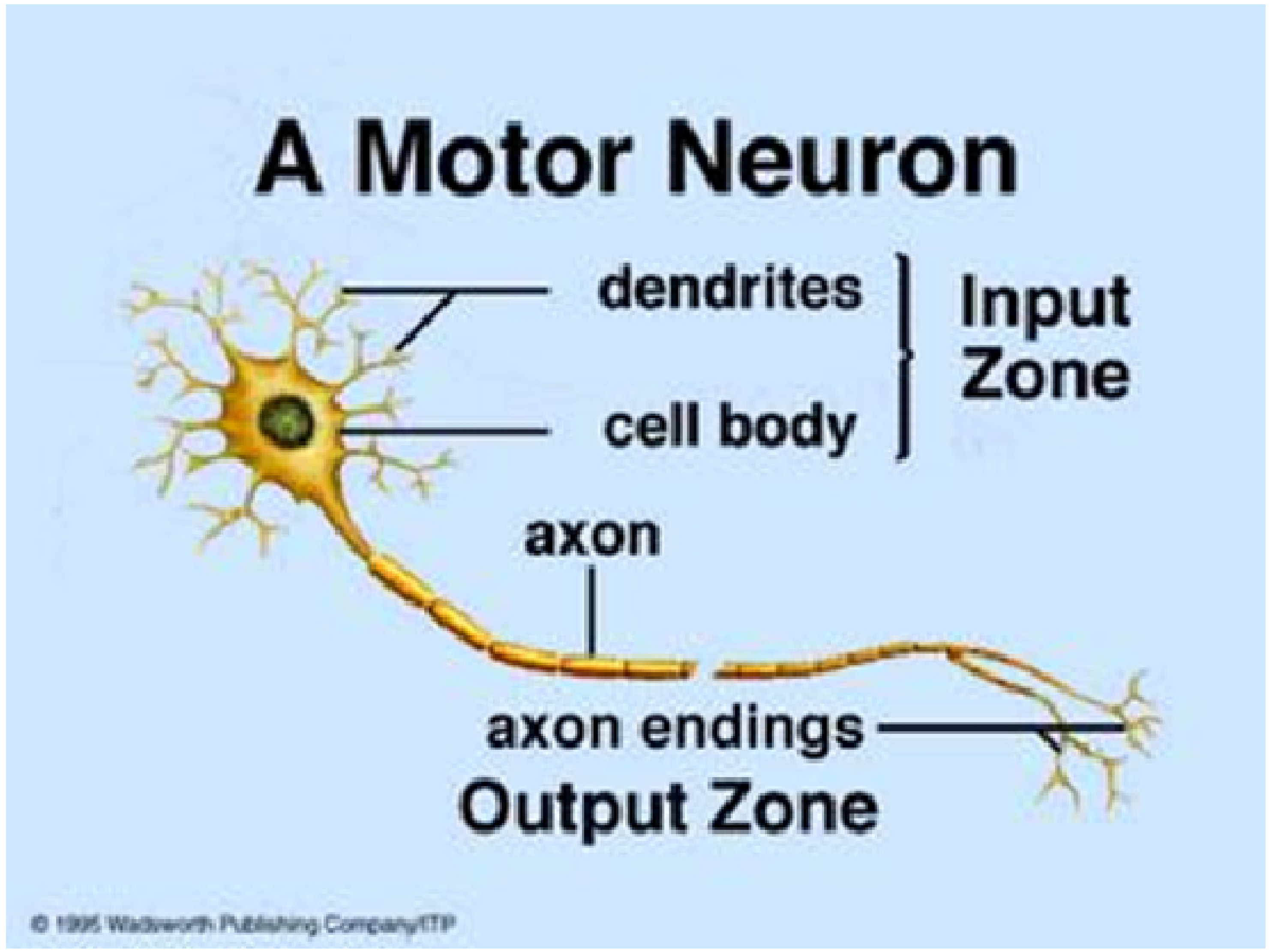


* Conventional computers use an algorithmic approach, but neural networks works similar to human brain and learns by example.


Axonal arborization
Axon from another cell
Synapse
Dendrite
Axon
Nucleus
Synapses
Cell body or Soma

Biological Neural Networks

- A biological neuron has three types of main components; dendrites, soma (or cell body) and axon.
- Dendrites receives signals from other neurons.



- The soma, sums the incoming signals. When sufficient input is received, the cell fires; that is it transmit a signal over its axon to other cells.

Artificial Neural network:

- ANN is an information processing system that has certain performance characteristics in common with biological nets.
- Several key features of the processing elements of ANN are suggested by the properties of biological neurons:

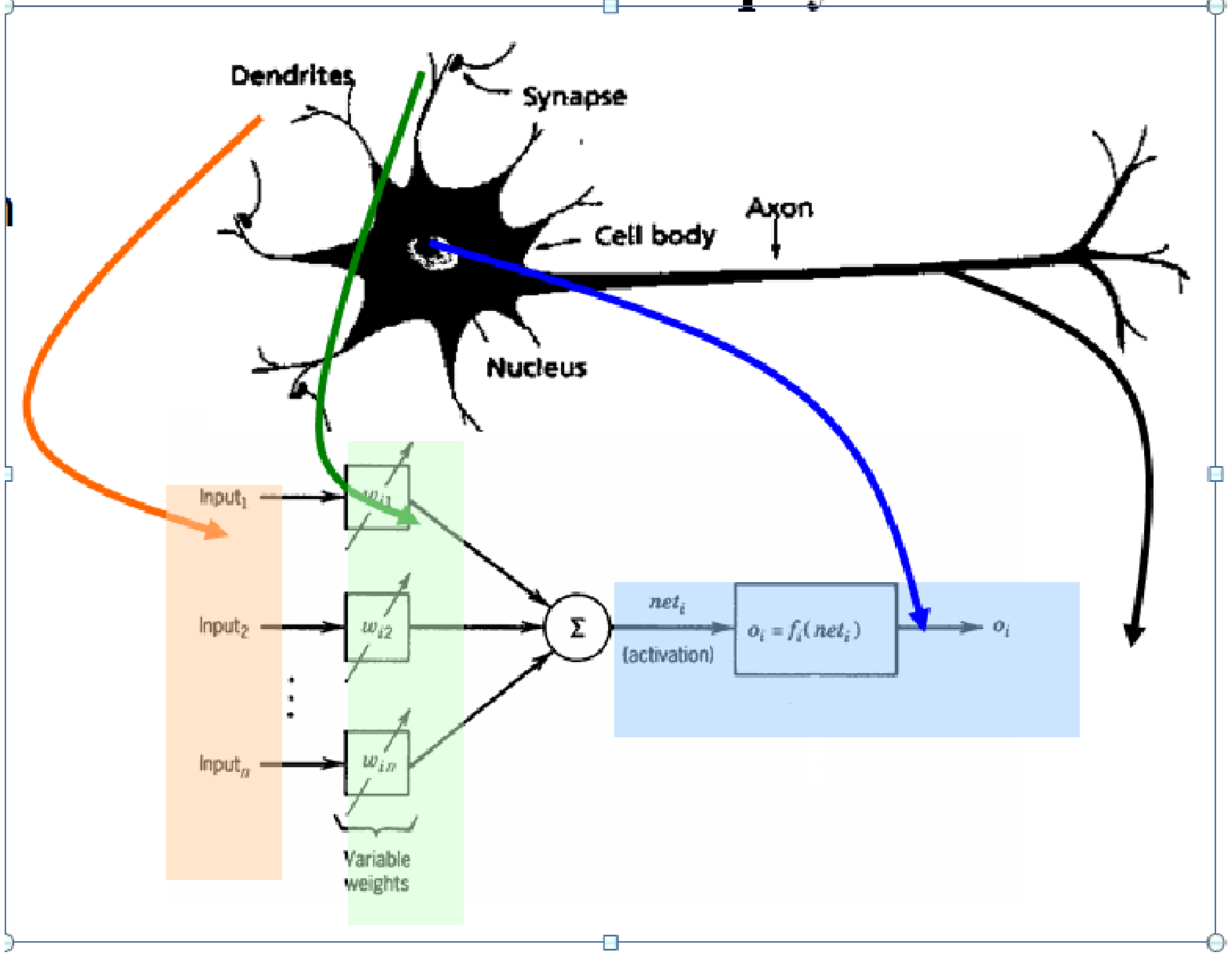
1. The processing element receives many signals.
2. Signals may be modified by a weight at the receiving synapse.
3. The processing element sums the weighted inputs.
4. Under appropriate circumstances (sufficient input), the neuron transmits a single output.
5. The output from a particular neuron may go to many other neurons.

From experience: examples / training data

Strength of connection between the neurons is stored as a weight-value for the specific connection.

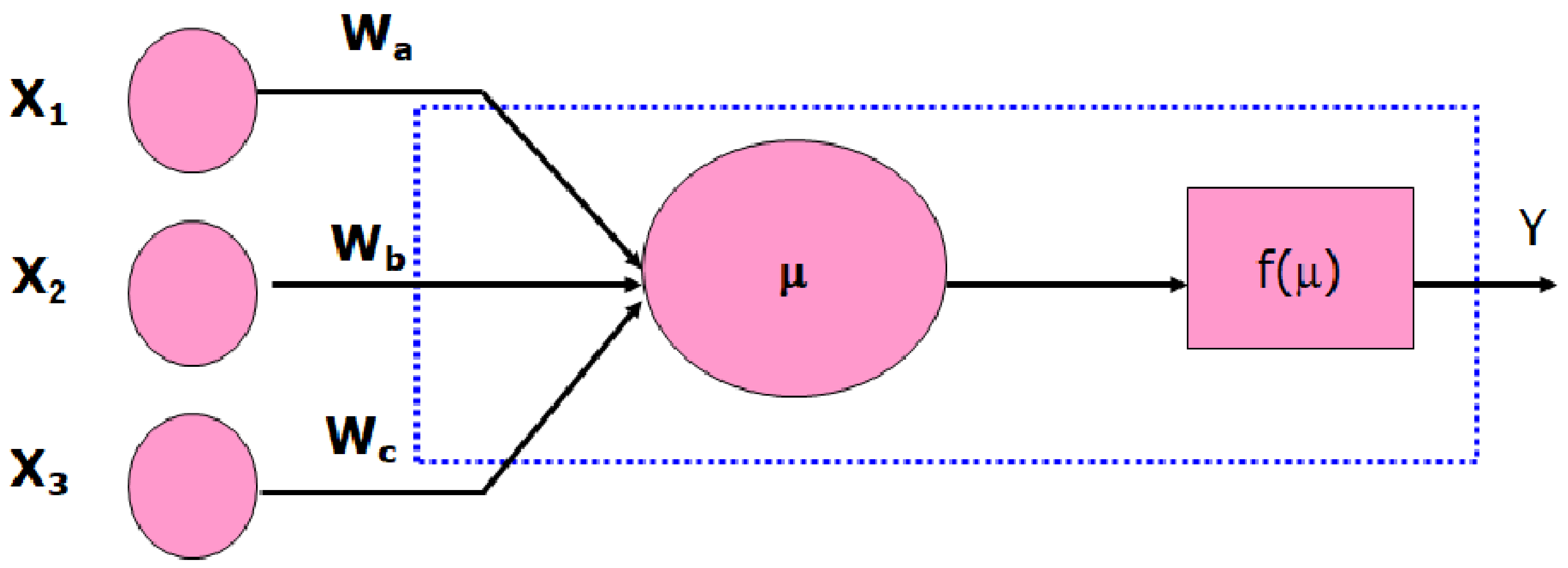
Learning the solution to a problem = changing the connection weights

A physical neuron



An artificial neuron

Model Of A Neuron



Input units (dendrite)

Connection weights (synapse)

Summing function

computation

$\underbrace{\hspace{6em}}_{\text{(soma)}}$

(axon)

- A neural net consists of a large number of simple processing elements called neurons, units, cells or nodes.

- Each neuron is connected to other neurons by means of directed communication links, each with associated weight.

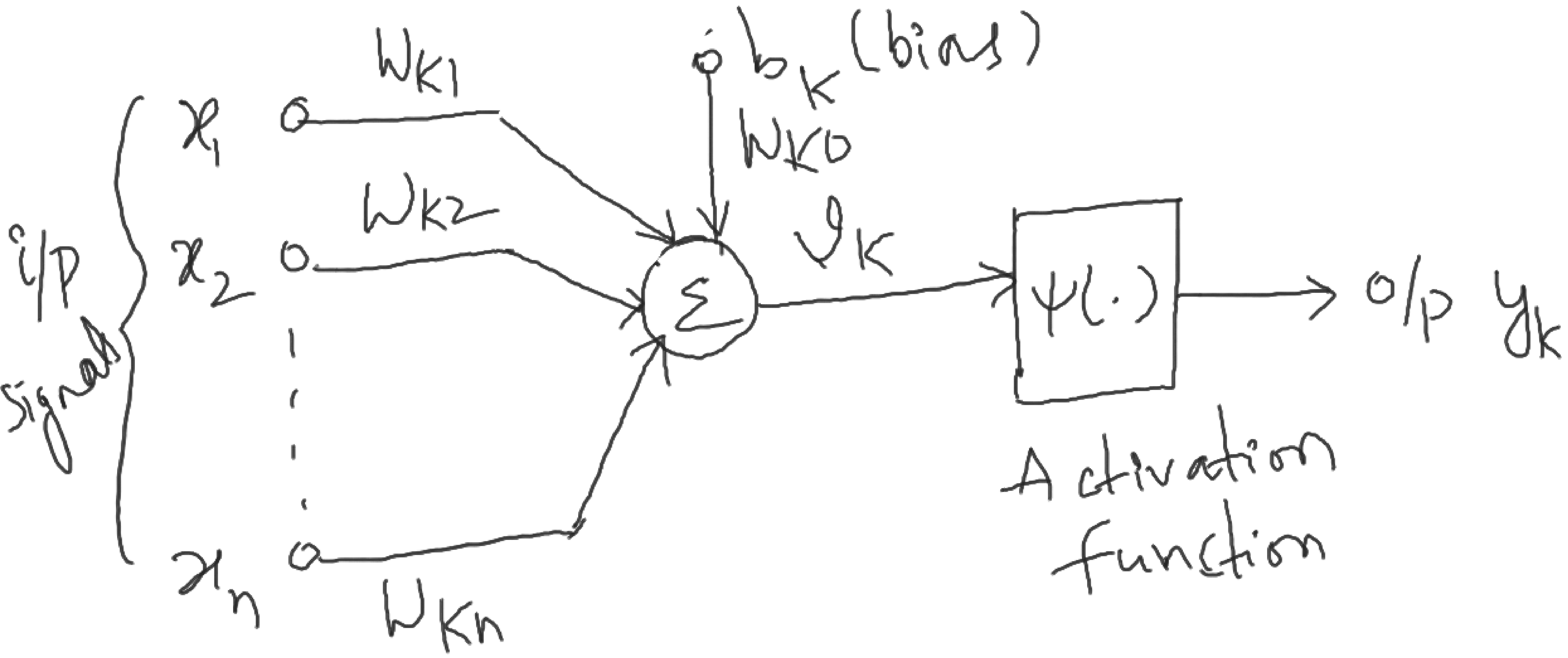
- The weight represent information being used by the net to solve a problem.

- Each neuron has an internal state, called its activation or activity level, which is a function of the inputs it has received. Typically, a neuron sends its activation as a signal to several other neurons.

- It is important to note that a neuron can send only one signal at a time, although that signal is broadcast to several other neurons.

- Neural networks are configured for a specific application, such as pattern recognition or data classification, through a learning process
- In a biological system, learning involves adjustments to the synaptic connections between neurons

➔ same for artificial neural networks (ANNs)



*Bias has the effect of increasing or lowering the net input of the activation function, depending on whether it is positive or negative respectively.

W_{kj} → a signal x_j at the input of synapse j connected to neuron k is multipli-ed by the synaptic weight W_{kj}

$$u_k = \sum_{j=1}^{n} W_{kj} x_j$$

$$y_k = \psi(u_k + b_k)$$

$$y_k = \psi(\vartheta_k), \quad \vartheta_k = u_k + b_k$$

$\psi(\cdot) \rightarrow$ Activation function or Squash function

$x_1, x_2, \cdots x_n \longrightarrow$ input signals

$w_{k0}, w_{k1}, w_{k2} \cdots w_{kn} \longrightarrow$ synaptic weights of k^{th} neuron

What are Activation functions and what are it uses in a Neural Network Model?

- *It is also known as **Transfer Function**. It can also be attached in between two Neural Networks.*
- Activation functions are important for a Artificial Neural Network to learn and understand the complex patterns.
- The main function of it is to introduce non-linear properties into the network.
- What it does is, it calculates the 'weighted sum' and adds direction and decides whether to 'fire' a particular neuron or not
- ***Their main purpose is to convert a input signal of a node in a A-NN to an output signal.*** That output signal now is used as a input in the next layer in the stack.
- The non linear activation function will help the model to understand the complexity and give accurate results.

The question arises that why can't we do it without activating the input signal?

- If we do not apply a Activation function then the output signal would simply be a simple ***linear function***.
- A *linear function* is just a polynomial of **one degree.** Now, a linear equation is easy to solve but they are limited in their complexity and have less power to learn complex functional mappings from data.
- A Neural Network without Activation function would simply be a **Linear regression Model,** which has limited power and does not performs good most of the times.
- *Also without activation function our Neural network would not be able to learn and model other complicated kinds of data such as images, videos , audio , speech etc.*

Types of Activation Functions

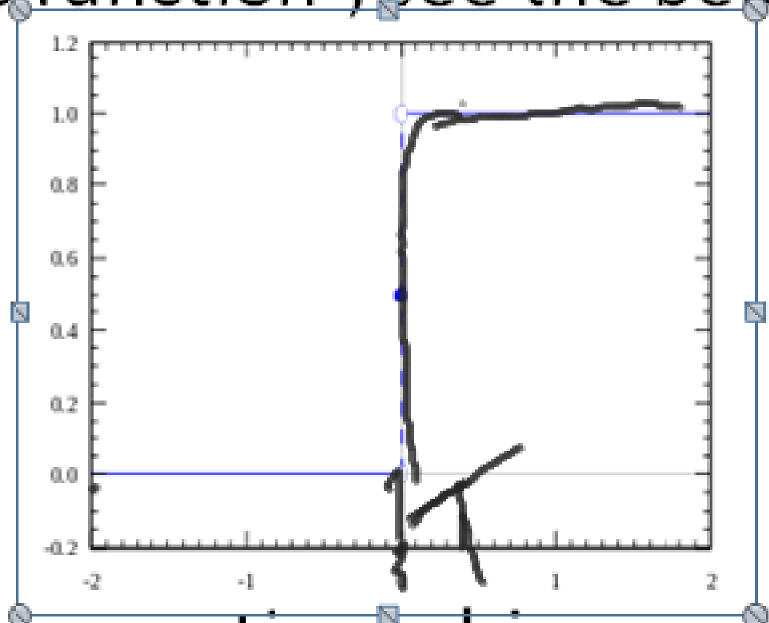
The most famous activation functions are given below,

1. Binary step
2. Linear
3. ReLU
4. LeakyReLU
5. Sigmoid
6. Tanh
7. Softmax

Step function

- Activation function A = “activated” if Y > threshold else not
- Alternatively, A = 1 if Y > threshold, 0 otherwise
- Well, what we just did is a “step function”, see the below figure.

$\varphi(v_k) T$



- **DRAWBACK:** Suppose you are creating a binary classifier. Something which should say a “yes” or “no” (activate or not activate). A Step function could do that for you! That’s exactly what it does, say a 1 or 0. Now, think about the use case where you would want multiple such neurons to be connected to bring in more classes. Class1, class2, class3 etc. What will happen if more than 1 neuron is “activated”. All neurons will output a 1 (from step function). Now what would you decide? Which class is it? Hard, complicated.

v_k

$y_k = \varphi(v_k) =$

$v_k > T$

$y_k = 1$

$v_k < T$

$y_k = 0$

Linear function

- A = cx
- A straight line function where activation is proportional to input (which is the weighted sum from neuron).
- This way, it gives a range of activations, so it is not binary activation. We can definitely connect a few neurons together and if more than 1 fires, we could take the max and decide based on that. So that is ok too. Then what is the problem with this?
- A = cx, derivative with respect to x is c. That means, the gradient has no relationship with X. It is a constant gradient and the descent is going to be on constant gradient. If there is an error in prediction, the changes made by back propagation is constant and not depending on the change in input.

$y = mx$

$\frac{dy}{dx} = m$

θ_k

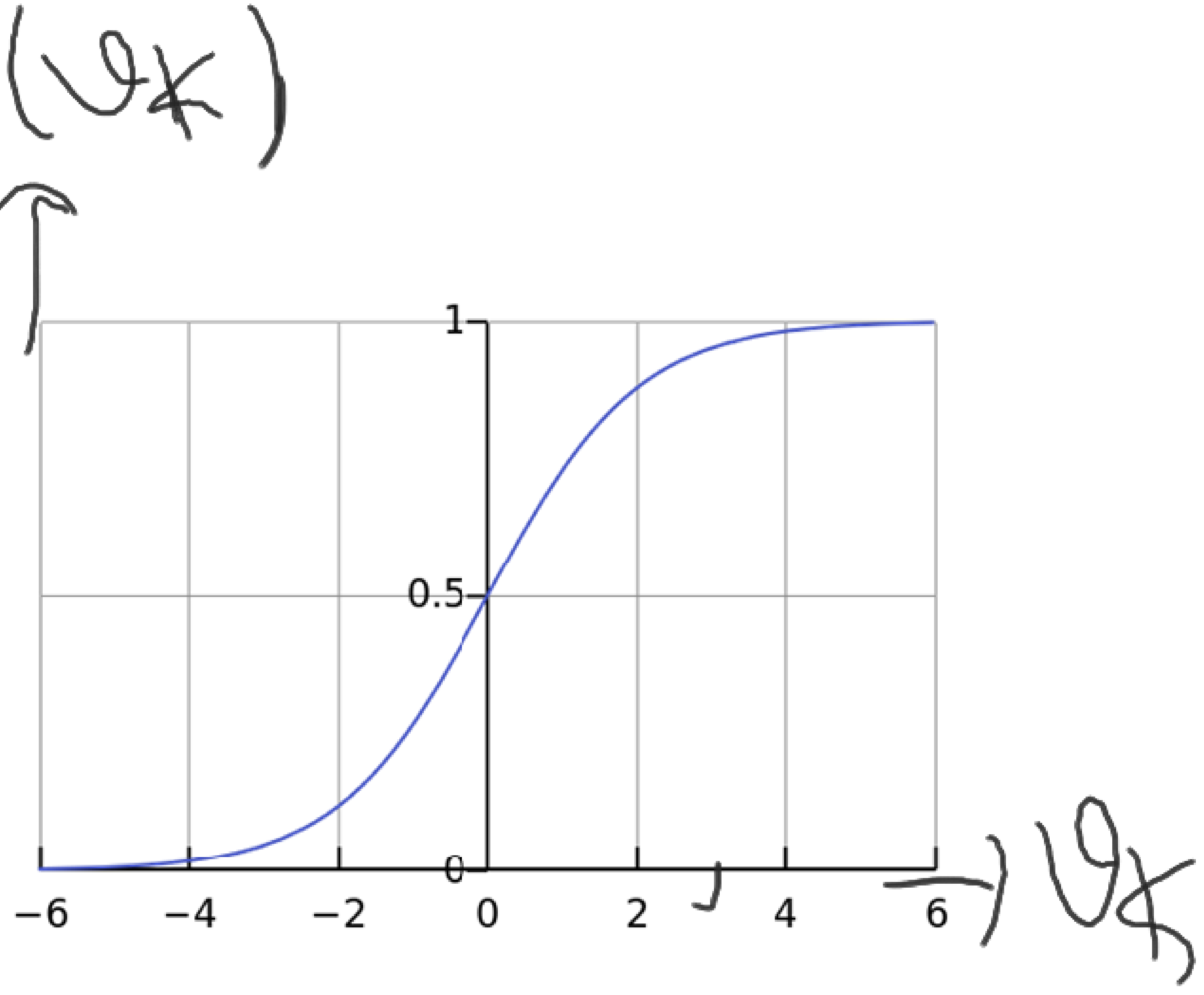
$y_k = \underline{c\,\theta_k}$

constant

Sigmoid function

$$\varphi(v) \quad A = \frac{1}{1+e^{-x}} \quad x = v_k$$

This looks smooth and "step function like". What are the benefits of this? It is nonlinear in nature. Combinations of this function are also nonlinear! Great. Now we can stack layers. What about non binary activations? Yes, that too! It will give an analog activation unlike step function. It has a smooth gradient too.



And if you notice, between X values -2 to 2, Y values are very steep. Which means, any small changes in the values of X in that region will cause values of Y to change significantly. That means this function has a tendency to bring the Y values to either end of the curve.

$$y_k = \frac{1}{1+e^{-av_k}}$$

- Looks like it's good for a classifier considering its property? Yes ! It tends to bring the activations to either side of the curve (above x = 2 and below x = -2 for example). Making clear distinctions on prediction.
- Another advantage of this activation function is, unlike linear function, the output of the activation function is always going to be in range (0,1) compared to (-inf, inf) of linear function. So we have our activations bound in a range. It won't blow up the activations then. This is great.
- Sigmoid functions are one of the most widely used activation functions today. Then what are the problems with this?
- If you notice, towards either end of the sigmoid function, the Y values tend to respond very less to changes in X. What does that mean? The gradient at that region is going to be small. It gives rise to a problem of "vanishing gradients". So what happens when the activations reach near the "near-horizontal" part of the curve on either sides?
- Gradient is small or has vanished (cannot make significant change because of the extremely small value). The network refuses to learn further or is drastically slow. There are ways to work around this problem and sigmoid is still very popular in classification problems.

Tanh Function

- Another activation function that is used is the tanh function.

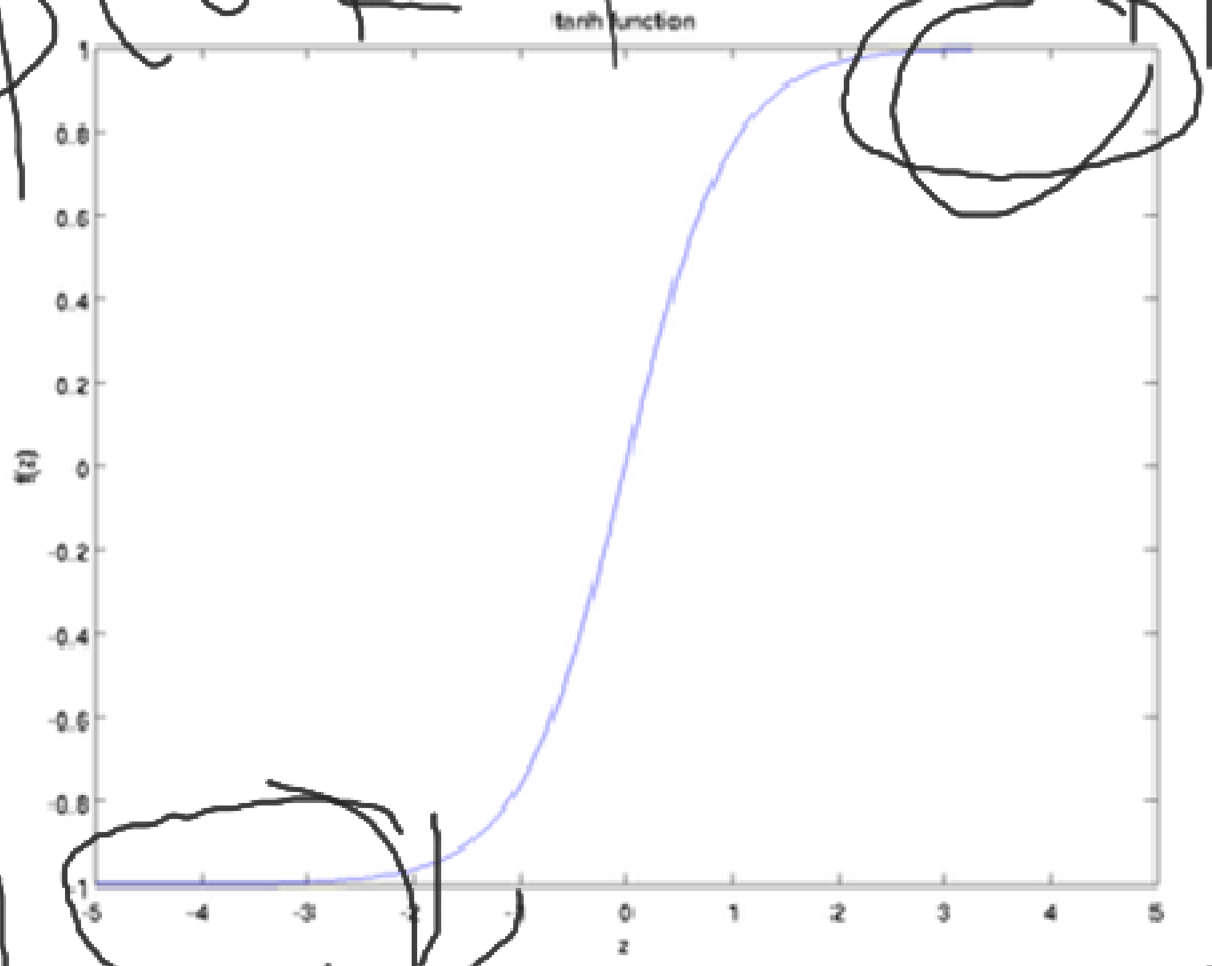
$\varphi(v_k)$

$$\boxed{f(x)} = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

This looks very similar to sigmoid. In fact, it is a scaled sigmoid function!

$$\tanh(x) = 2\,sigmoid(2x) - 1$$

$\varphi(v) \to +1$



$+1$

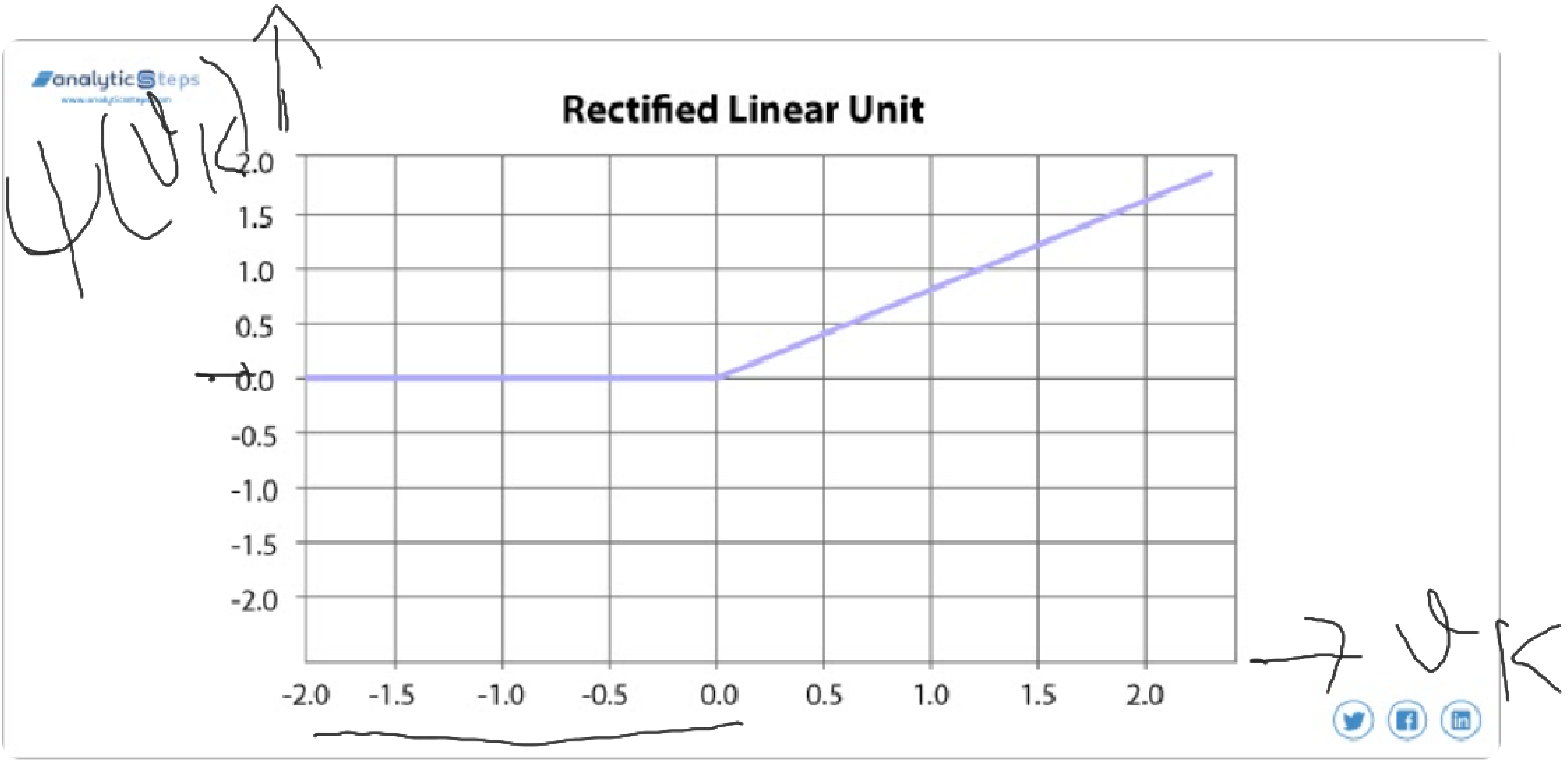
-1

$\rightarrow v_k$

$x = v_k$

- This has characteristics similar to sigmoid that we discussed above. It is nonlinear in nature, so great we can stack layers! It is bound to range (-1, 1) so no worries of activations blowing up. One point to mention is that the gradient is stronger for tanh than sigmoid (derivatives are steeper). Deciding between the sigmoid or tanh will depend on your requirement of gradient strength. Like sigmoid, tanh also has the vanishing gradient problem.
- Tanh is also a very popular and widely used activation function. Especially in time series data.

Rectified linear unit or ReLU is most widely used activation function right now which ranges from **0 to infinity**, All the negative values are converted into zero, and this conversion rate is so fast that neither it can map nor fit into data properly which creates a problem, but where there is a problem there is a solution.

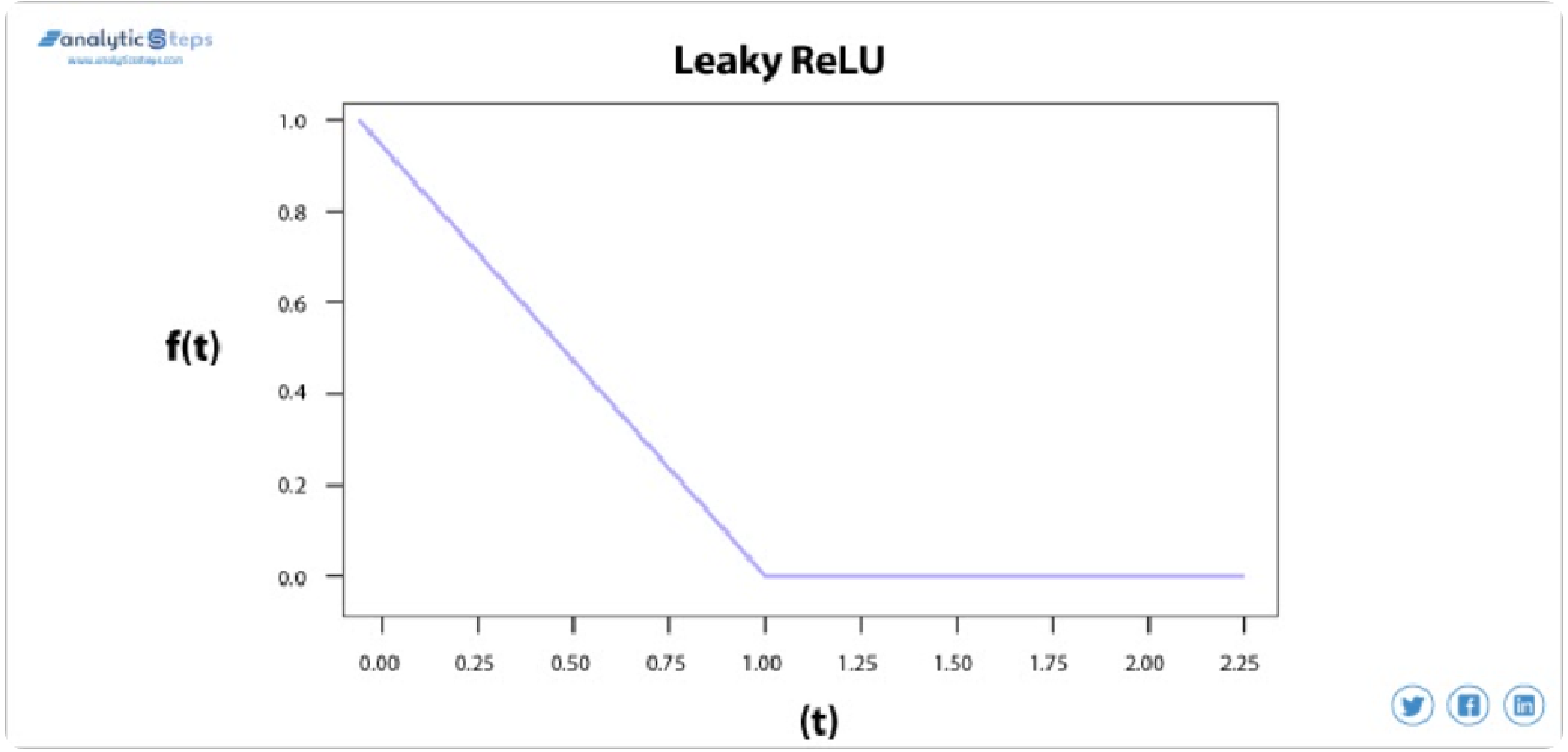


Rectified Linear Unit activation function

- At first look, this would look like having the same problems of linear function, as it is linear in positive axis. First of all, ReLu is nonlinear in nature. And combinations of ReLu are also non linear! (in fact it is a good approximator. Any function can be approximated with combinations of ReLu). Great, so this means we can stack layers. It is not bound though. The range of ReLu is [0, inf). This means it can blow up the activation.
- Another point to discuss here is the sparsity of the activation. Imagine a big neural network with a lot of neurons. Using a sigmoid or tanh will cause almost all neurons to fire in an analog way. That means almost all activations will be processed to describe the output of a network. In other words, the activation is dense. This is costly. We would ideally want a few neurons in the network to not activate and thereby making the activations sparse and efficient.
- ReLu give us this benefit. Imagine a network with random initialized weights (or normalized) and almost 50% of the network yields 0 activation because of the characteristic of ReLu (output 0 for negative values of x). This means a fewer neurons are firing (sparse activation) and the network is lighter. ReLu seems to be awesome! Yes it is, but nothing is flawless.. Not even ReLu.

- Because of the horizontal line in ReLu (for negative X), the gradient can go towards 0. For activations in that region of ReLu, gradient will be 0 because of which the weights will not get adjusted during descent. That means, those neurons which go into that state will stop responding to variations in error/ input (simply because gradient is 0, nothing changes). This is called **dying ReLu problem**. This problem can cause several neurons to just die and not respond making a substantial part of the network passive. There are variations in ReLu to mitigate this issue by simply making the horizontal line into non-horizontal component . For example, $y = 0.01x$ for $x<0$ will make it a slightly inclined line rather than horizontal line. This is leaky ReLu. There are other variations too. The main idea is to let the gradient be non zero and recover during training eventually.
- ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. That is a good point to consider when we are designing deep neural nets.

4. Leaky ReLU Activation Function

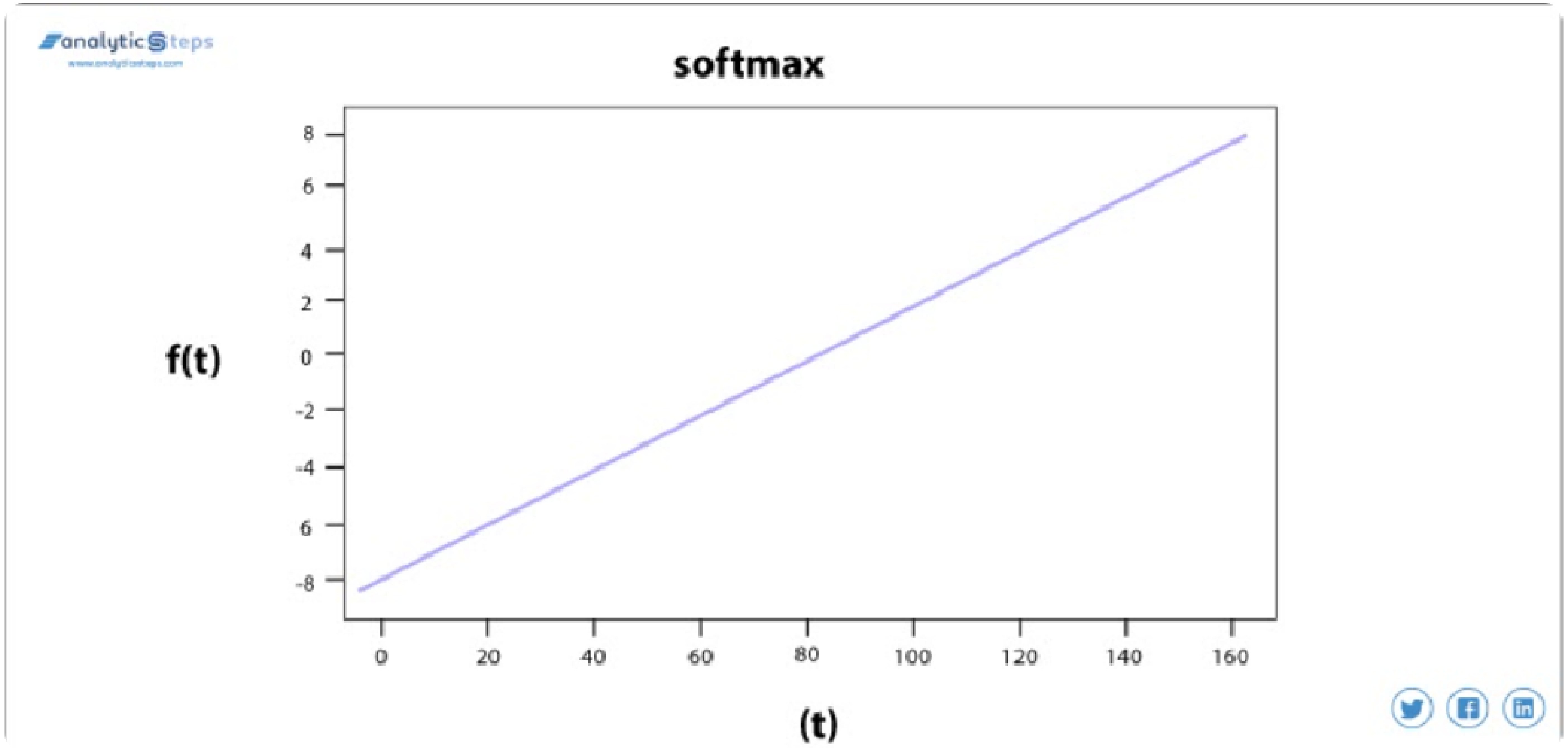


Leaky ReLU Activation Function

We needed the **Leaky ReLU activation** function to solve the ***'Dying ReLU'*** problem, as discussed in ReLU, we observe that all the negative input values turn into zero very quickly and in the case of Leaky ReLU we do not make all negative inputs to zero but to a value near to zero which solves the major issue of ReLU activation function.

7. Softmax Activation Function

Softmax is used mainly at the last layer i.e output layer for decision making the same as sigmoid activation works, the softmax basically gives value to the input variable according to their weight and the sum of these weights is eventually one.



Softmax on Binary Classification

For **Binary classification**, both **sigmoid**, as well as **softmax**, are equally approachable but in case of multi-class classification problem we generally use softmax and cross-entropy along with it.

:

Stochastic model of a neuron

-- The neuron previously discussed is the deterministic in that its input -output behaviur is precisely defined for all inputs.

--For some applications of neural networks stochastic neuron model is required.

--The decision for a neuron to fire is probabilistic.

--Let x denotes state of the neuron, and p(v) denote the probability of firing, where v is the induced field of the neuron.

$$x = \begin{cases} +1, & \text{with probability } P(v) \\ -1, & \text{with probability } 1 - P(v) \end{cases}$$

$$P(\vartheta) = \frac{1}{1 + e^{(-\vartheta/T)}}$$

$T \rightarrow$ Pseudo temperature

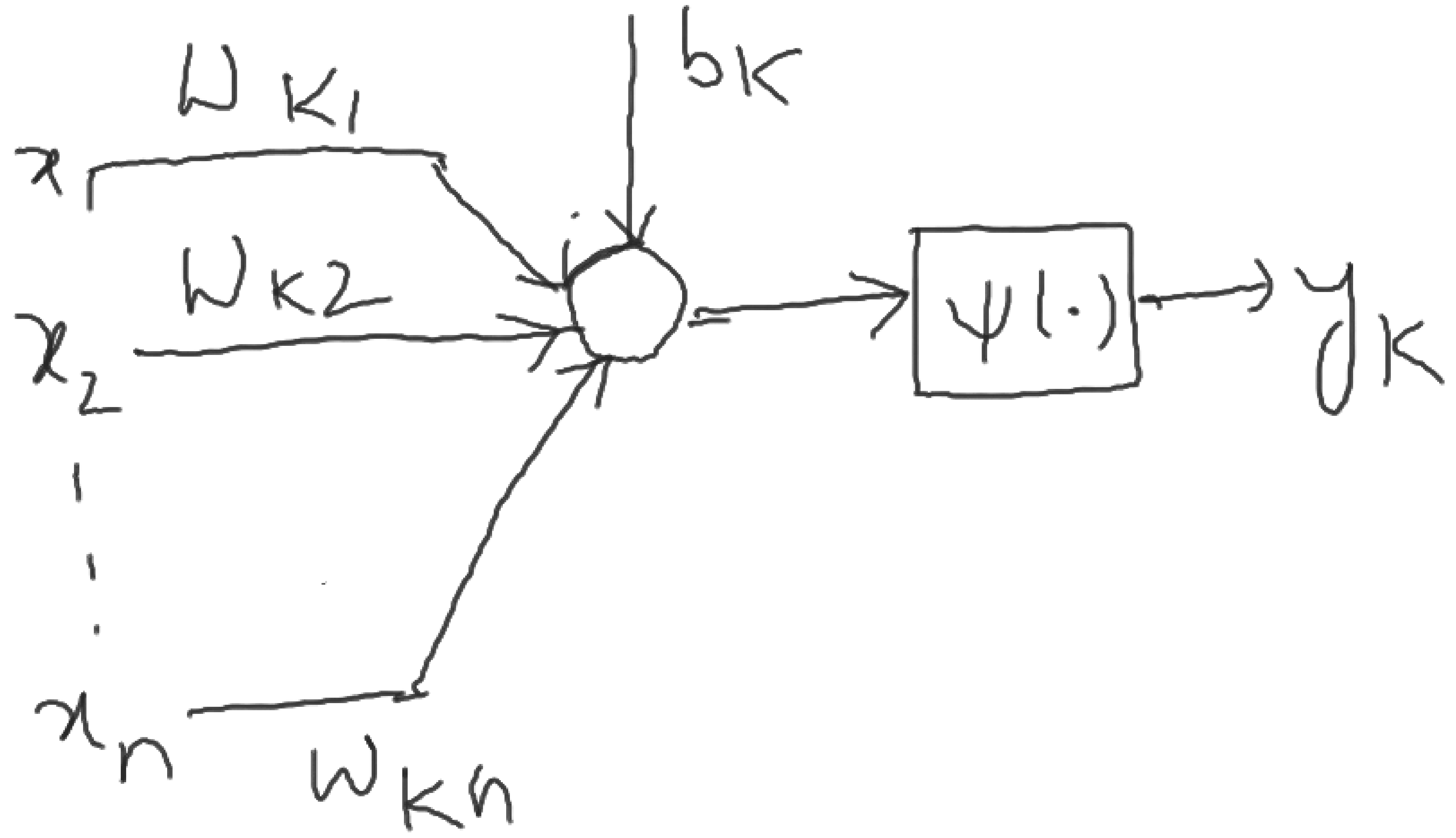
$\rightarrow$ T is used to control noise level and therefore uncertainity in firing

Neural networks viewed as directed graphs:

The block diagram provides functional description of the various elements that constitue the model of an artificial neuron.

A signal flow graph is a network of directed links(brantches) that are interconnected at certain points called nodes.

The flow of signals in the various parts of the graph is dictated by three basic rules.



Neural Networks Viewed as Directed Graph

- **Figures 5 and 7 are *functional description* of a neuron. We can use *signal-flow graph* *(a network of directed links that are interconnected to points called nodes)* to describe the network using these rules:**
  - *A signal flow along a link only in the direction defined by the arrow on the link*
  - *A node signal equal the algebraic sum of all signals entering the node.*
  - *The signal at a node is transmitted to each outgoing link originated from that node.*

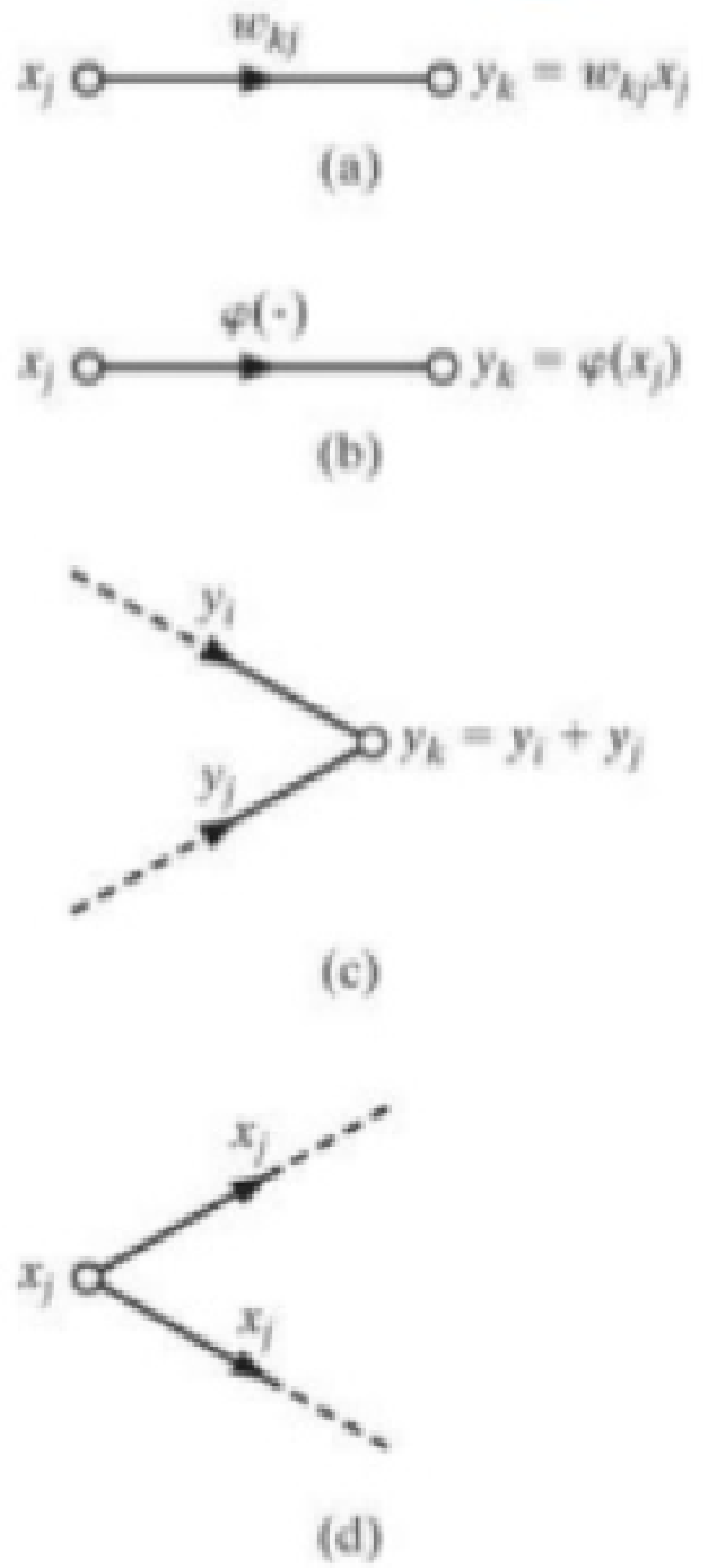


Figure 9 illustrating basic rules for the construction of signal-flow graphs.

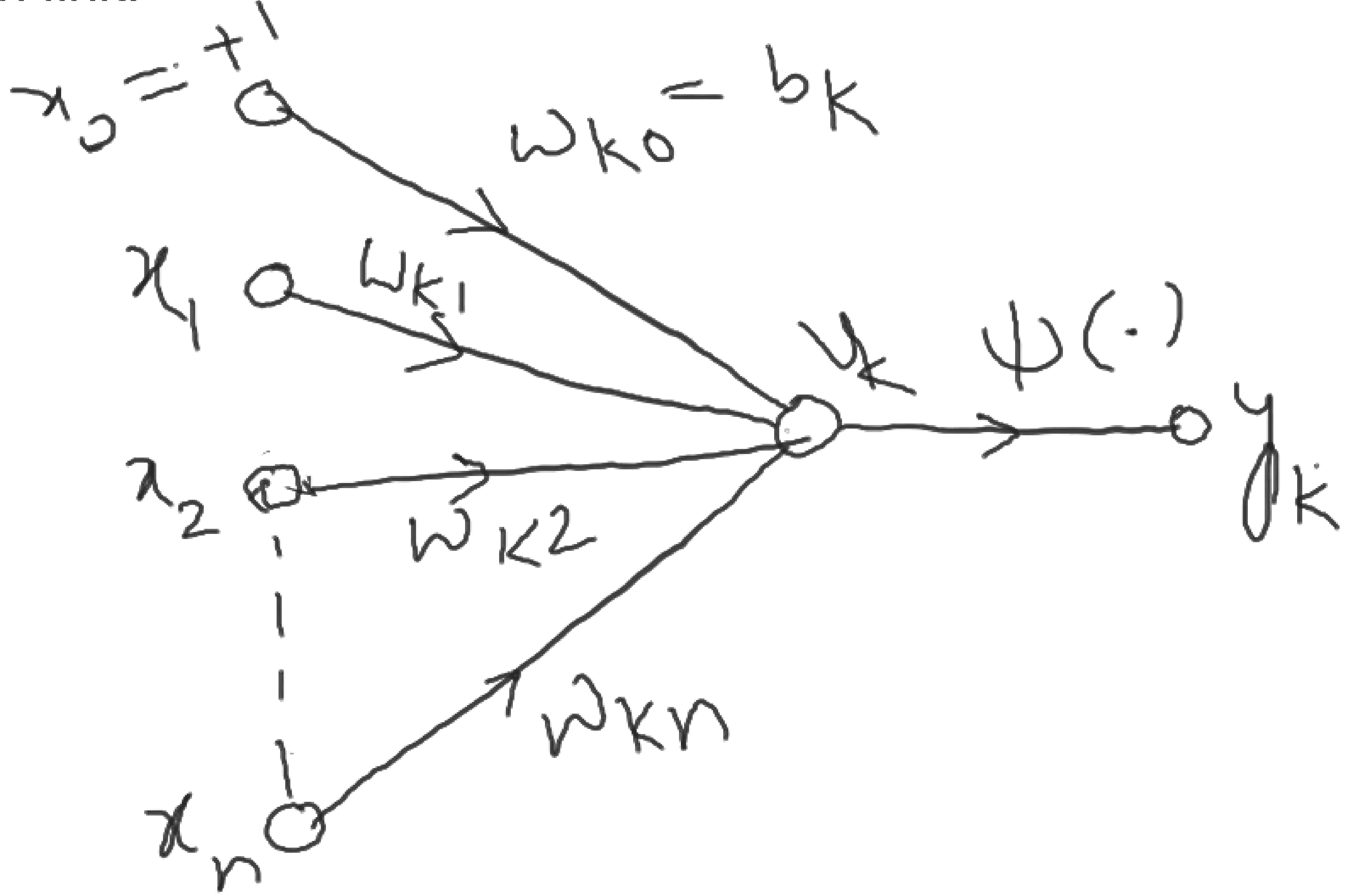
A neural network is a directed graph consisting of nodes with interconnecting synaptic and activation links and is characterised by four properties.

1) Each neuron is represented by a set of linear synaptic links, an externally applied bias and possibly non linear activation link.

2) The synaptic links of a neuron weight their respective input signals.

3) The weighted sum of the input signals defines the induced local field of the neuron.

4) The activation link squashes the induced local fied of the neuron to produce an output.



A directed graph so defined is complete in the sense that it describes not only the signal flow from neuron to neuron, but also the signal flow inside each neuron.

The focus of attention restricted to signal flow from neuron to neuron , we may use a reduced form of this graph by omitting the details of signal flow inside the individual neuron. Such a directed graph is said to be partially complete.

It is characterized as follows

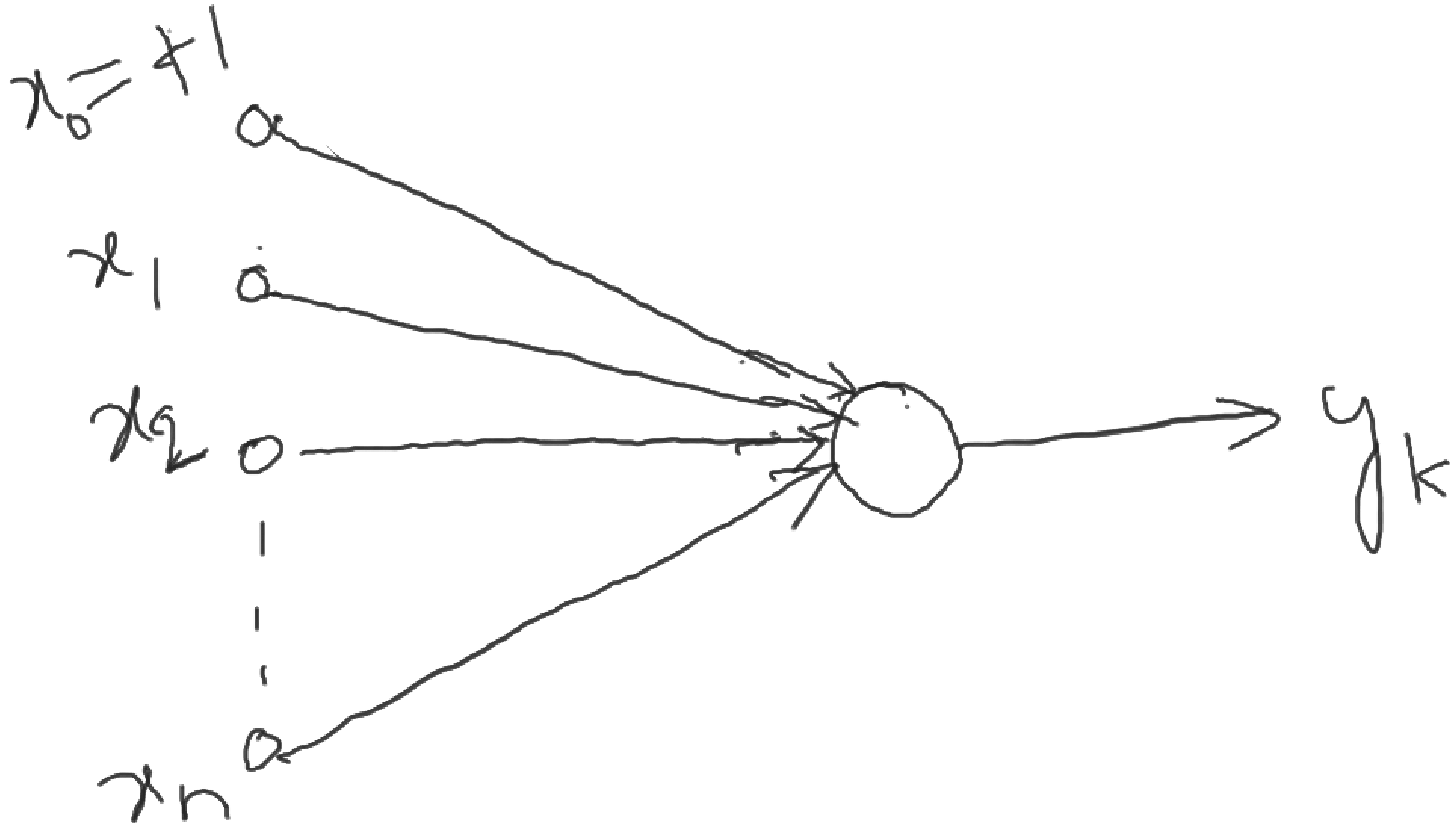
1)Source nodes supply input signals to the neurons.

2)Each neuron is represented by a single node called a computation node.

3) the commmunication links interconnecting the source and computation nodes of the graph carry no weight, they merely provide directions of signal flow in the graph.

A partially complete directed graph is referred to as an architectural graph.

Architectural graph of neuron:



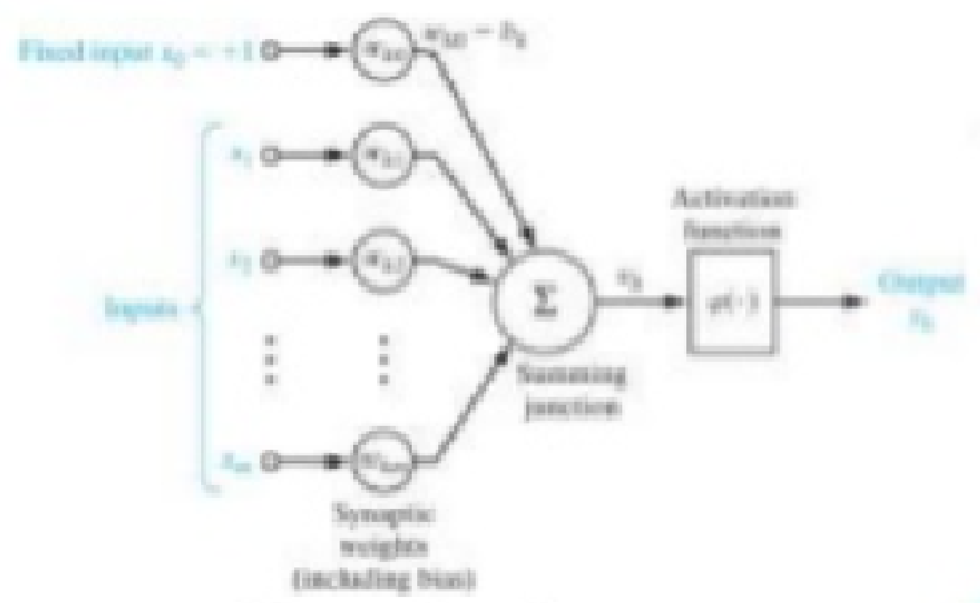
-- Block diagram providing functional description of the network.

--Signal flow graph , providing a complete description of the signal flow in the network.

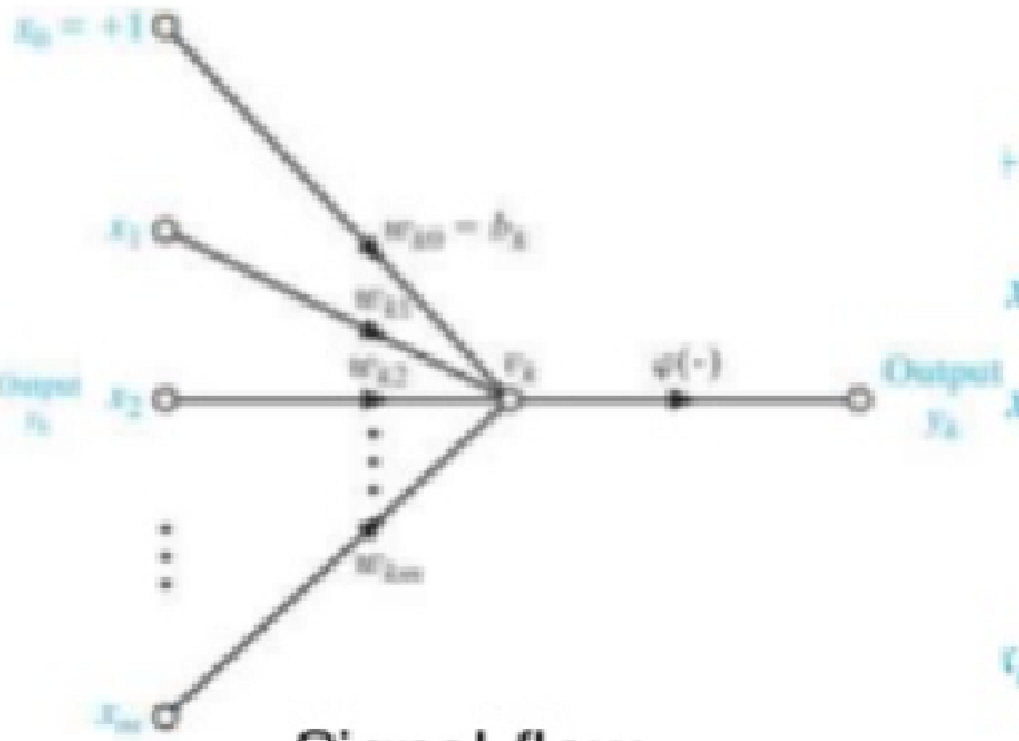
--Architectural graph describing the network layout.

Neural Networks Viewed as Directed Graph

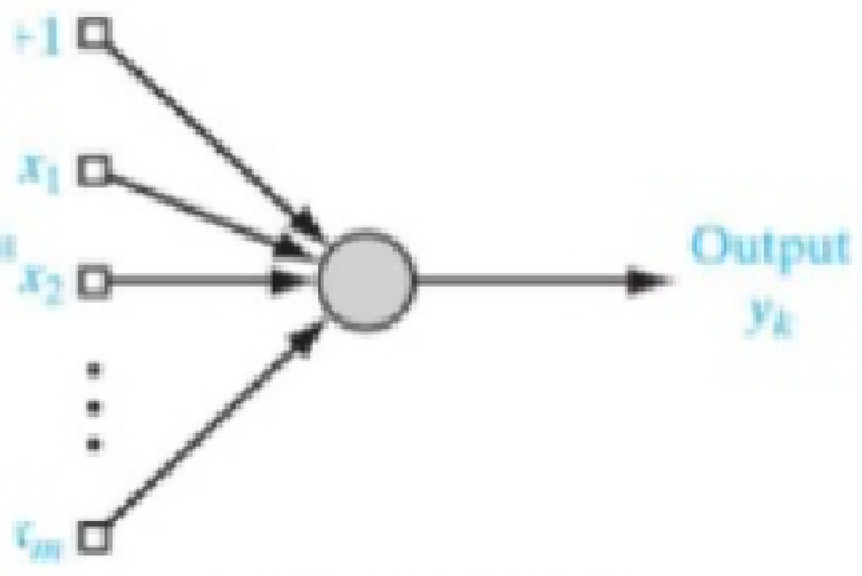
- **We have three graphical representations of a neural network.**
  - Block diagram (*Functional*)
  - *Complete* directed graph (*Signal-Flow*)
  - *Partially Complete* directed graph (*Architectural*)



Functional



Signal-flow



Architectural

Feedback

- ***Feedback*** **is said to exist in** ***dynamic*** **systems whenever the output of a node influences in part the input of that particular node. (Very important in the study of** ***recurrent networks*****.**

$$y_k(n) = \mathbf{A}[x'_j(n)]$$

and

$$x'_j(n) = x_j(n) + \mathbf{B}[y_k(n)]$$

$$\boxed{y_k(n) = \frac{\mathbf{A}}{1-\mathbf{AB}}[x_j(n)]}$$

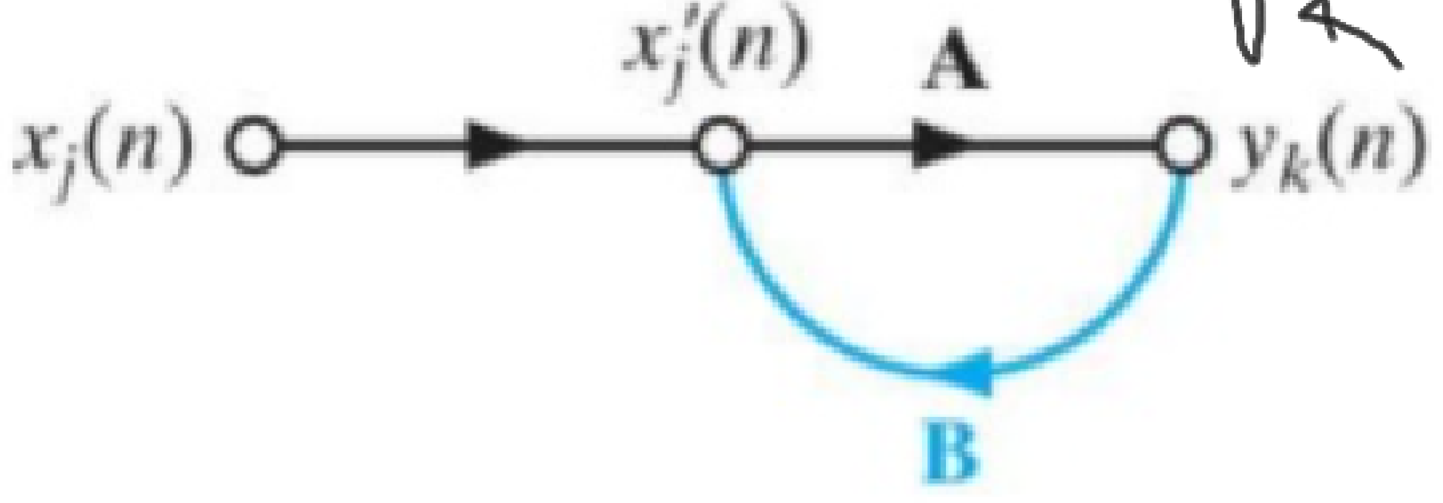


Figure 12 Signal-flow graph of a single-loop feedback system.

$y_k(n)$
$= A(x_j(n) + B y_k(n))$

$y_k(n) = A x_j(n)$
$+ AB y_k(n)$

$y_k(n)(1-AB) = A x_j(n$

A $\rightarrow$ fixed weight w $\qquad (1-x)^{-1}$

B $\rightarrow$ unit delay operator z^{-1}

$$\frac{A}{1-AB} = \frac{w}{1-wz^{-1}} = w(1-wz^{-1})^{-1} \qquad w(w^0 z^0 + wz^{-1} + w^2 z^{-2} + w^3 z^{-3})$$

$\rightarrow$ using the binomial expansion

$$\frac{A}{1-AB} = w \sum_{l=0}^{\infty} w^l z^{-l}$$

$$\therefore y_K(n) = \left(\sum_{l=0}^{\infty} w^{l+1} z^{-l} \right) x_j(n)$$

Network Architectures

- **Single-layer Feedforward Networks**
  - Note that: we do not count the input layer of the source nodes because no computation is performed there.

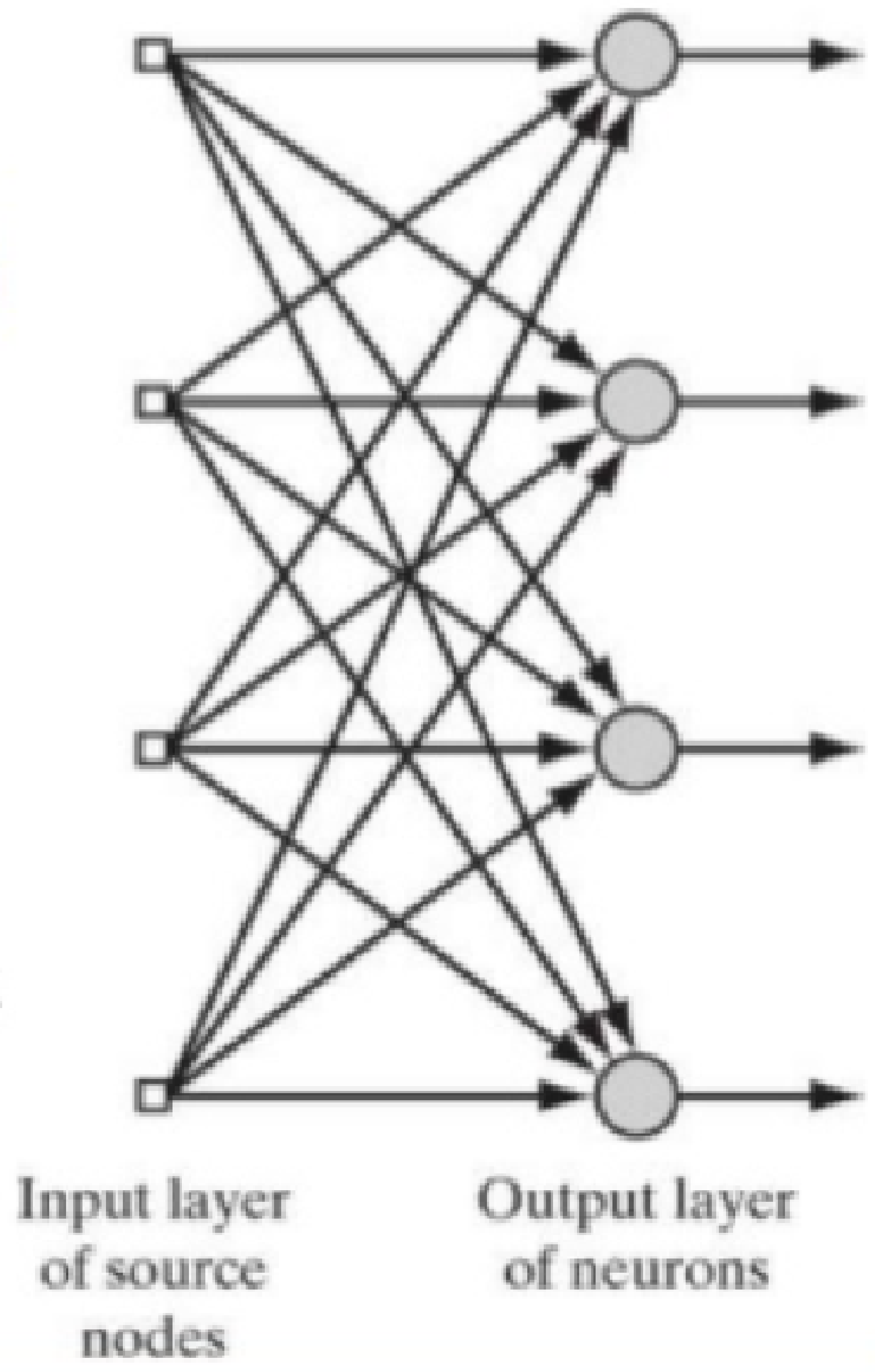


Figure 15 Feedforward network with a single layer of neurons.

Network Architectures

- **Multilayer Feedforward Networks**
  - Input layer (source nodes)
  - One or more hidden layers
  - One output layer
  - It is referred to as 10-4-2 network.
  - Could be *fully* or *partially* connected.

  - By adding one or more hidden layers, the network is enabled to extract higher order statistics from its input).

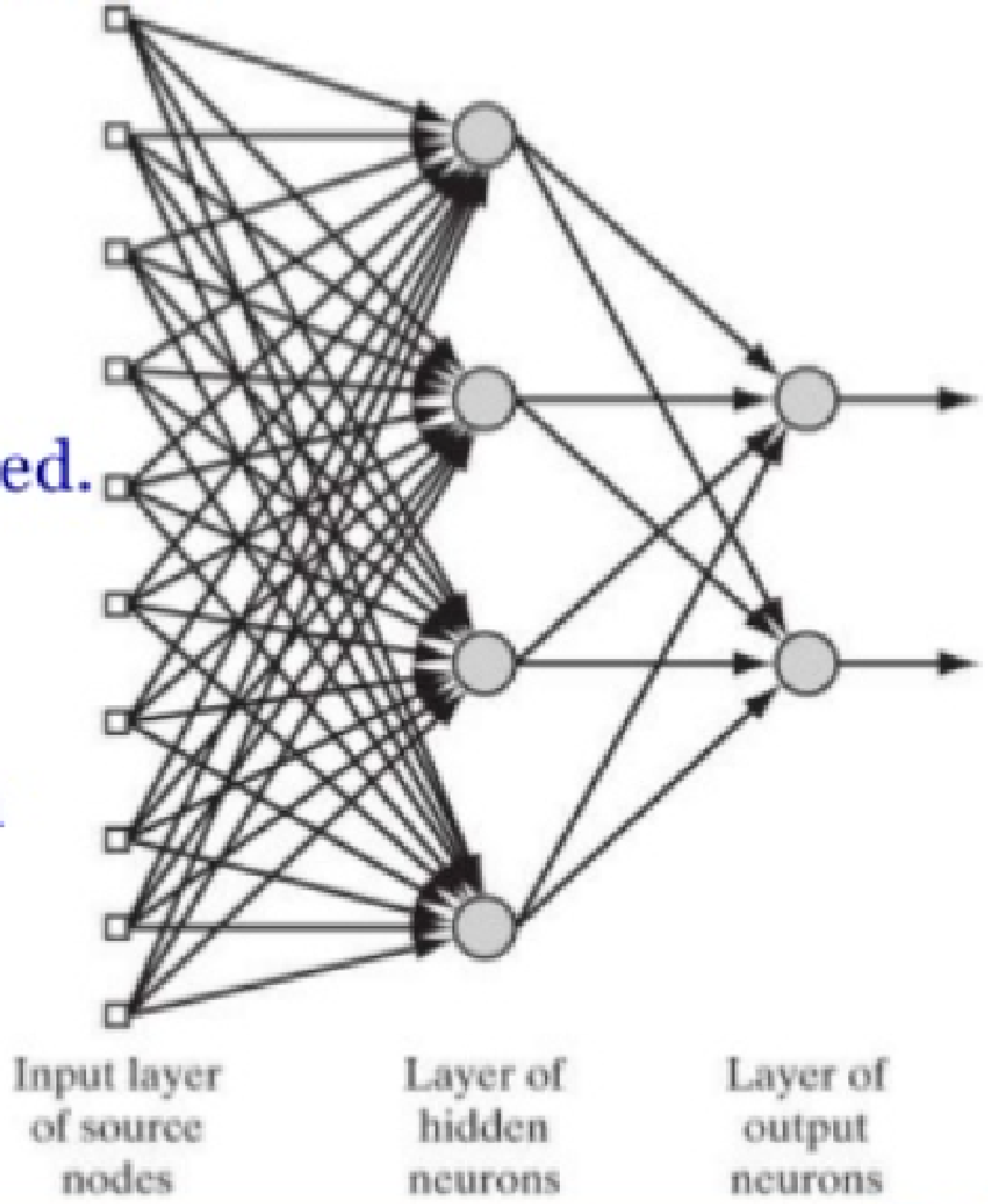
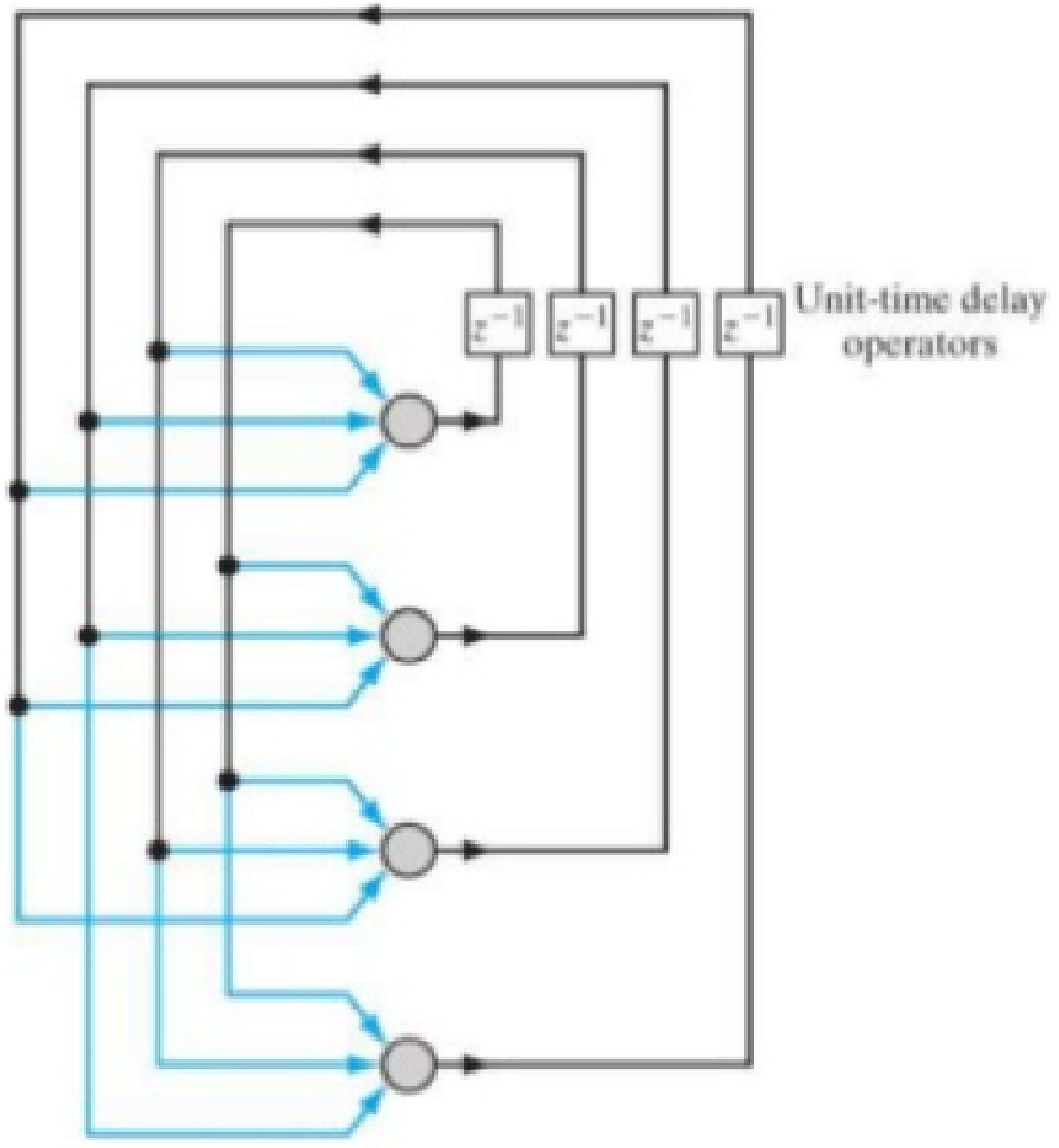


Figure 16 Fully connected feedforward network with one hidden layer and one output layer.

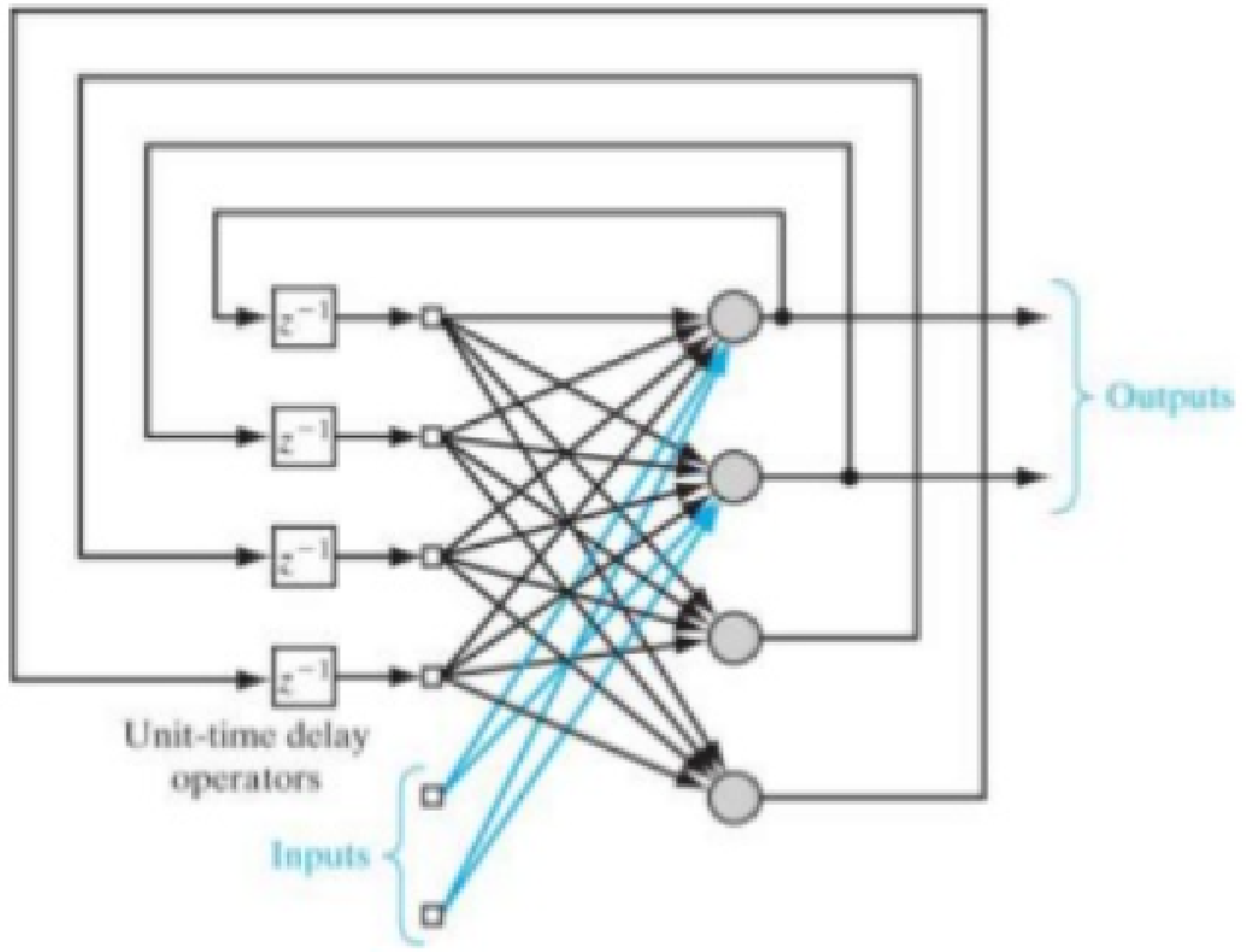
Network Architectures

- **Recurrent Networks**
  - A recurrent neural networks should has *at least* one feedback loop
  - *Self-feedback* refers to a situation where the output of a node is feedback into its own input.



Network Architectures

- **Recurrent Networks**
  - A recurrent neural networks with self feedback loop and with hidden neurons.



Knowledge Representation

- ***Knowledge* refers to**
    - *stored information or models used by a person or machine to interpret, predict, and approximately respond to the outside world.*
- **Characteristics of *knowledge representation:***
    - What information is actually made explicit?
    - How the information is physically encoded for subsequent use?

Knowledge Representation

- **The *major task* for a neural network is**
  - *to learn a model* of the world (environment) in which it is embedded, and *to maintain* the model sufficiently *consistently* with the real world so as *to achieve* the *specific goals* of the application of interest.

Knowledge of the world consists of Two kinds of information:

1) The known world state, represented by facts about what is and what has been known, this form of knowledgeis reffered to asprior information.

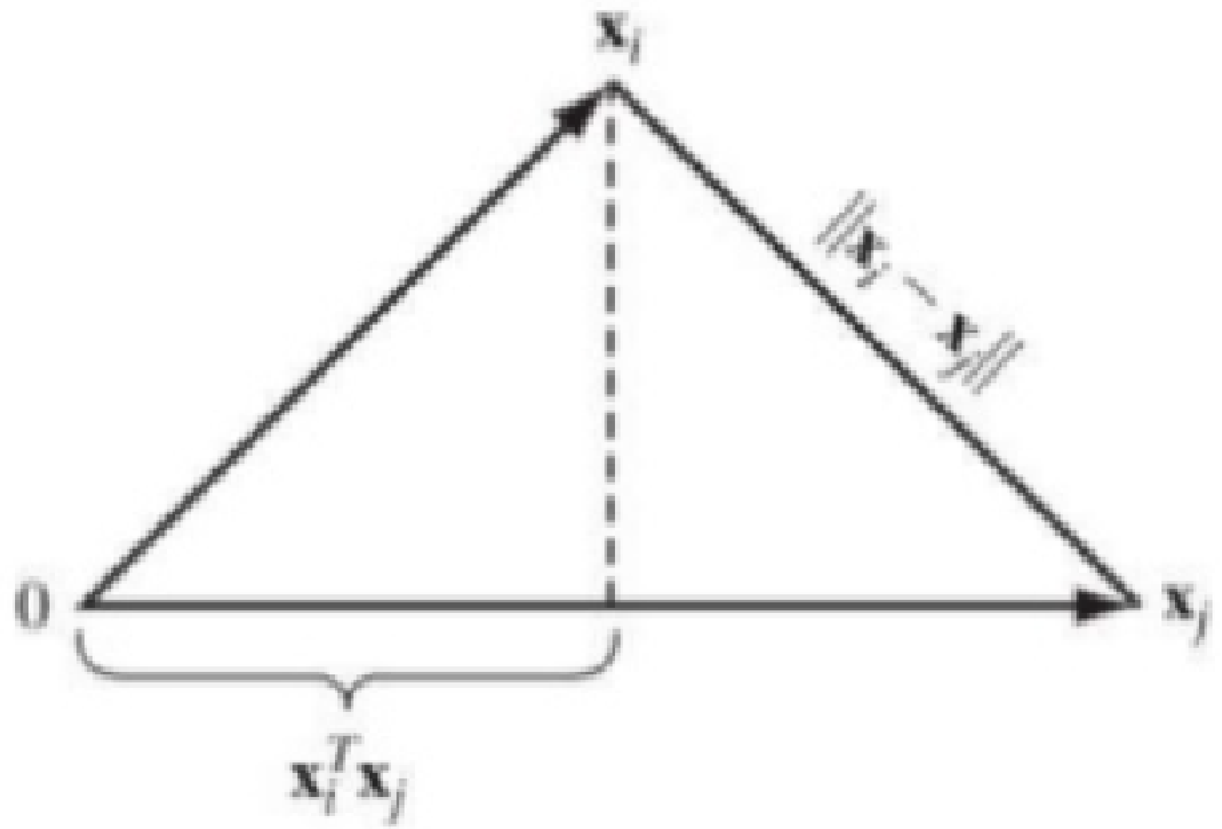
2)Observations (measurements) of the worl, obtained by means of sensors designed to probe the environment in which the neural network is supposed to operate.

The observations so obtained provide the pool of information from which the examples used to train the network are drawn.

Aset of input - output pairs, with each pair consisting of an input signal and the corresponding desired response, is referred as a set of training data or training samples.

Knowledge Representation

- **(Commonsense) Roles of Knowledge Representation**
- **Role 1: Similar inputs form similar classes should usually produce similar representation.**
- **Role 2: Items to be categorized as separate classes should be given widely different representation.**
- **Role 3: if a particular feature is important, then there should be a large number of neurons are involved in the representation.**
- **Role 4: Prior information and invariances should be built into the design of a NN when they are available.**



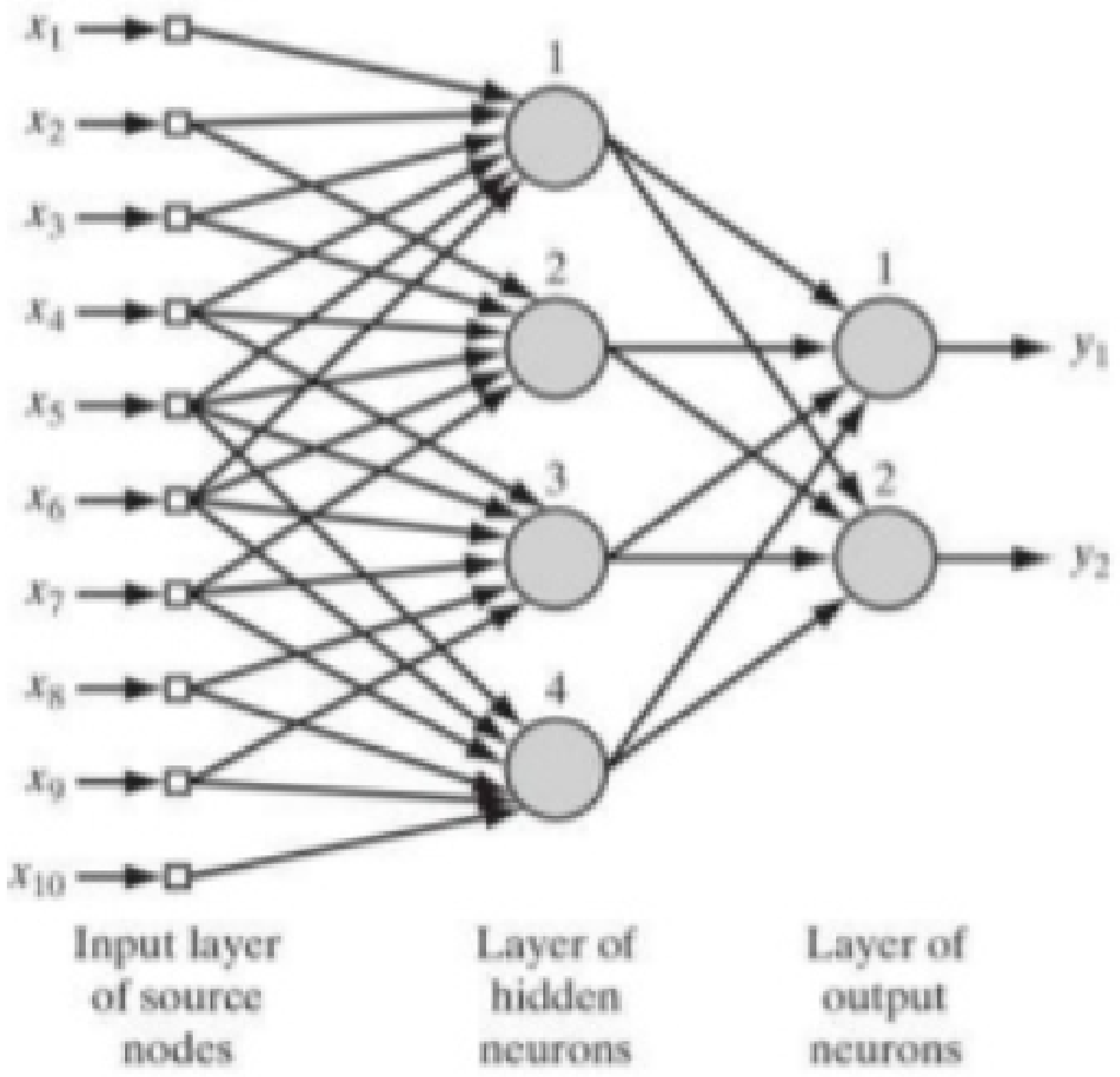
$\mathbf{x}_i$
$\|\mathbf{x}_i - \mathbf{x}_j\|$
0
$\mathbf{x}_j$
$\mathbf{x}_i^T \mathbf{x}_j$

Figure 19 Illustrating the *relationship* between *inner product* and *Euclidean distance* as measures of *similarity* between patterns.

Knowledge Representation

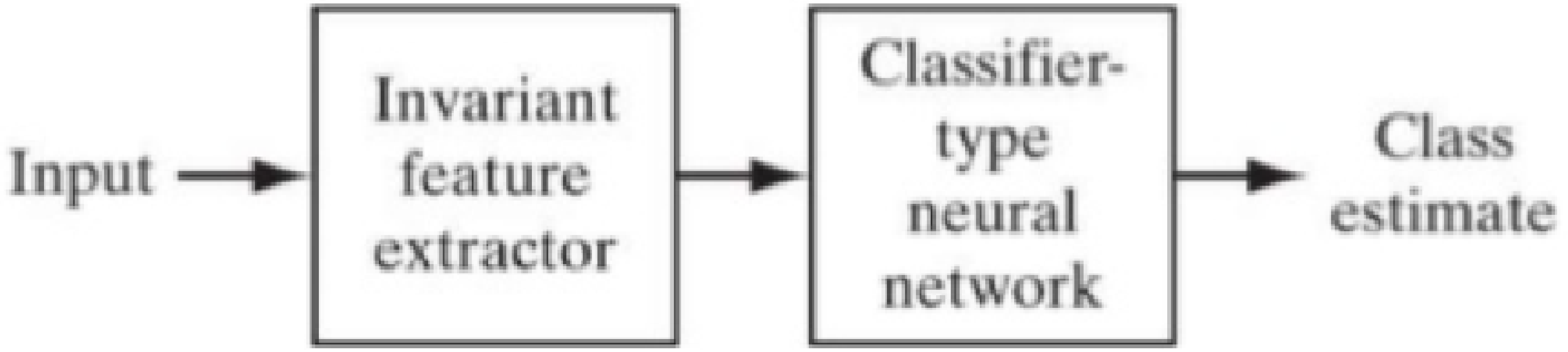
- **How to Build prior Info into Neural Network Design?**

- Currently ,there are no well-defined rules. But rather some *ad hoc* procedures:
  - Restricting the network *architecture*, which is achieved through the use of *local connections* (*receptive fields*).
  - Constraining the *choice* of *synaptic weights*, which is implemented through the use of *weight-sharing*.
- NNs that make use of these two techniques are called *Convolutional Networks*.



Knowledge Representation

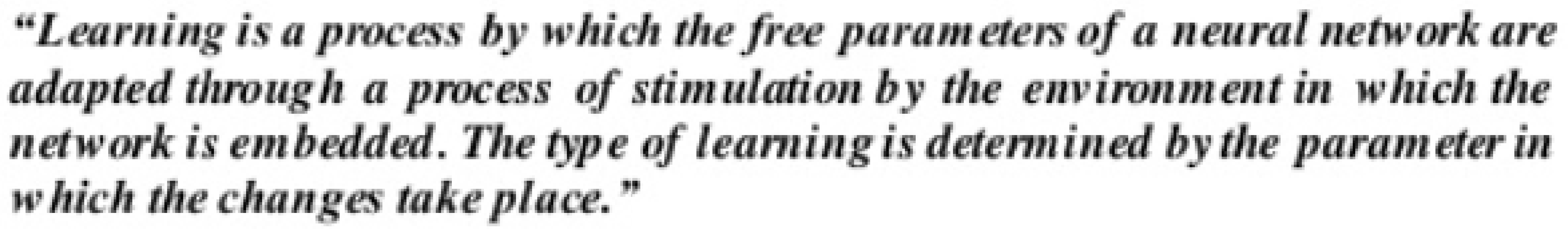
- **How to Build Invariance into Neural Network Design?**
  - Invariance by *Structure*
    - For example, enforcing in-plain rotation by making $w_{ij} = w_{ji}$
  - Invariance by *Training*
    - Including different examples of each class.
  - Invariant *Feature Space*
    - Transforming the data to invariant feature space (Fig. 21).



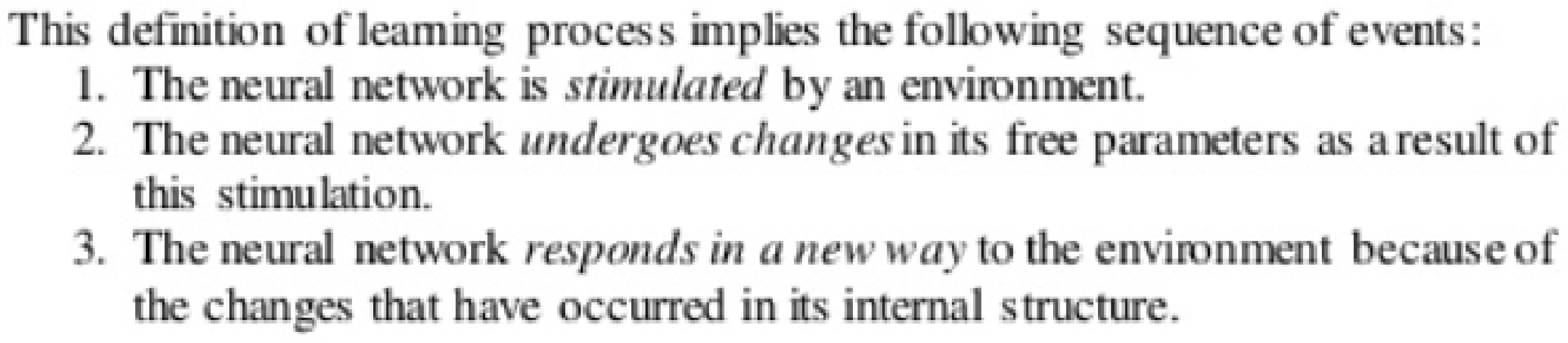
Learning Process:

---One of the most important properties of neural network is to improve their performance by taking into account the past experience. This is achieved through process called learning.

---A neural network learns about its environment through an interactive process of adjustments applied to its synaptic weights and bias levels. Ideally, the network becomes more knowledgable about its environment after each iteration of the learning process.



"Learning is a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded. The type of learning is determined by the parameter in which the changes take place."


This definition of learning process implies the following sequence of events:

1. The neural network is *stimulated* by an environment.
2. The neural network *undergoes changes* in its free parameters as a result of this stimulation.
3. The neural network *responds in a new way* to the environment because of the changes that have occurred in its internal structure.

Learning paradigms:

There are two major learning paradigms

1) Supervised learning

2) Unsupervised learning

Supervised learning:

---Supervised learning, sometimes referred to as learning with a teacher is a learning technique that sets parameters of an artificial neural network from training data which serves as the teacher.

---The task of the learning artificial neural network is to set the values of its parameters for any valid input value after having seen output value.

---The training data set consists of labeled pairs of input and desired output value that is traditionally represented in data vectors.

--- as close as possibleThe neural network, after learning from training data or teacher provides an output to a random input which resembles the training data example.

---The deviation from actual output is called the error signal.

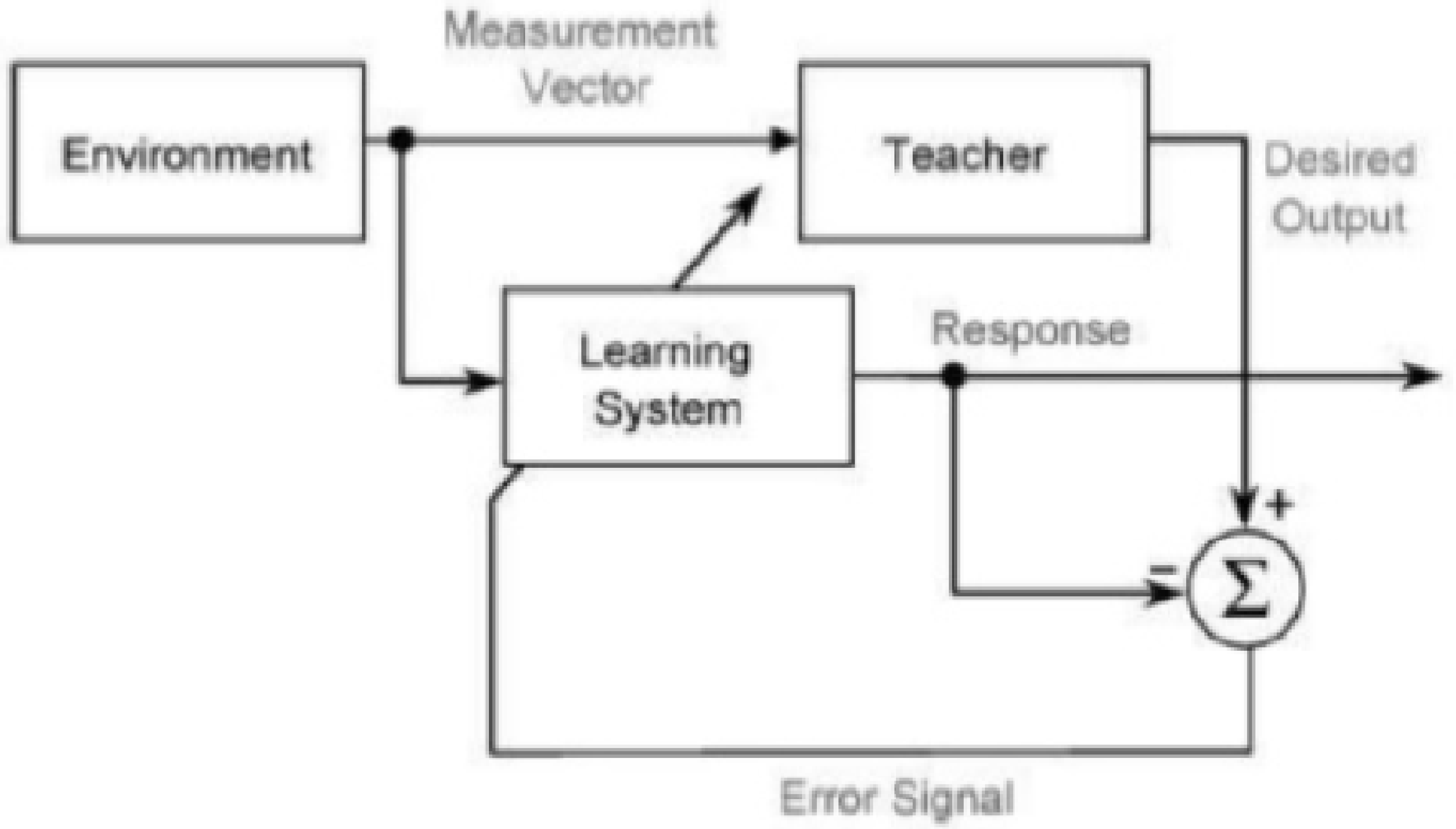


Fig. 1: Block diagram of Supervised Learning Model

Unsupervised learning:

---The unsupervised training model consists of the environment, represented by measurement vector.

---The measurement vector is fed to the learning system and the system response is obtained. Based upon the system response and the adaptation rule employed, the weights of the learning system are adjusted to obtain te desired performance.

---Unlike supervised learning, the unsupervised learning does not need a desired output for each input feature vector.

---The adaptation rule in the unsupervised training algorithm performs the error signal generation role the teacher performs in supervised learning.

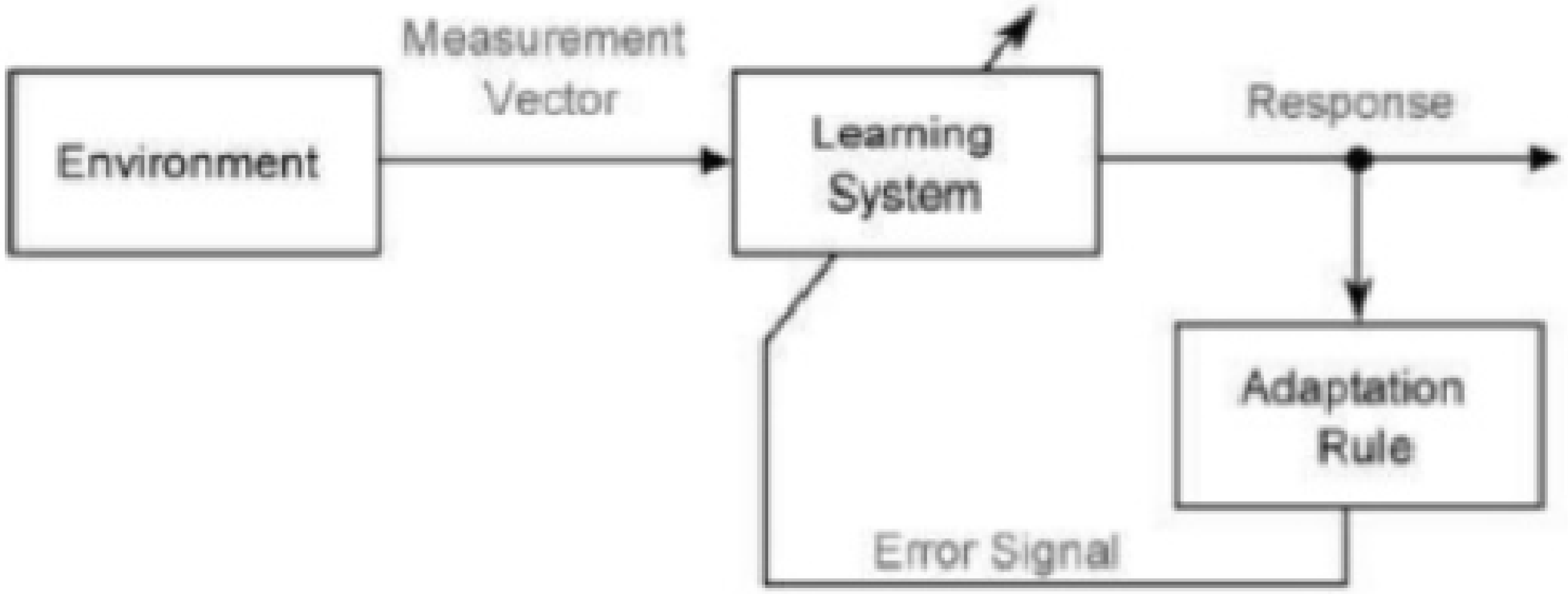
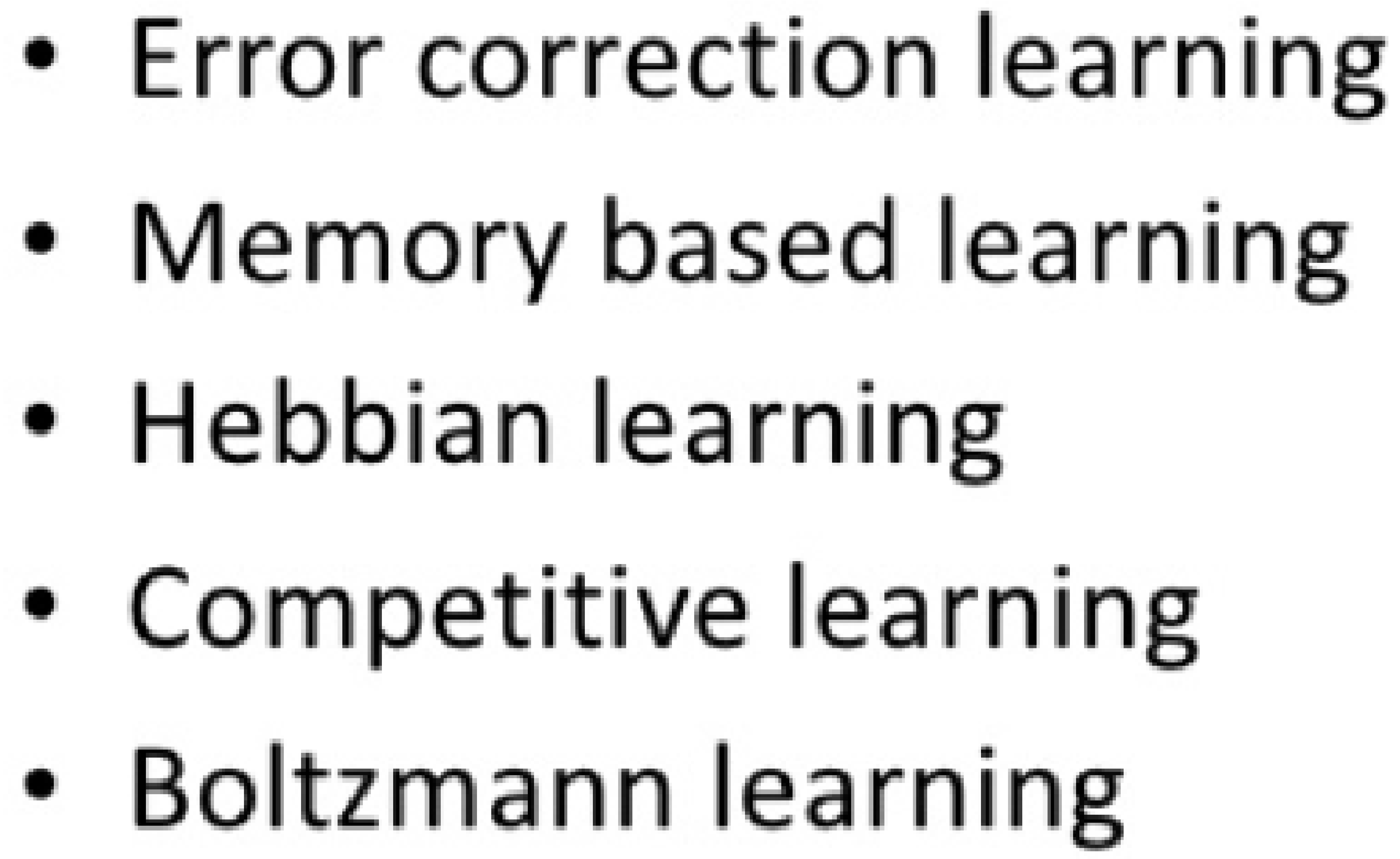


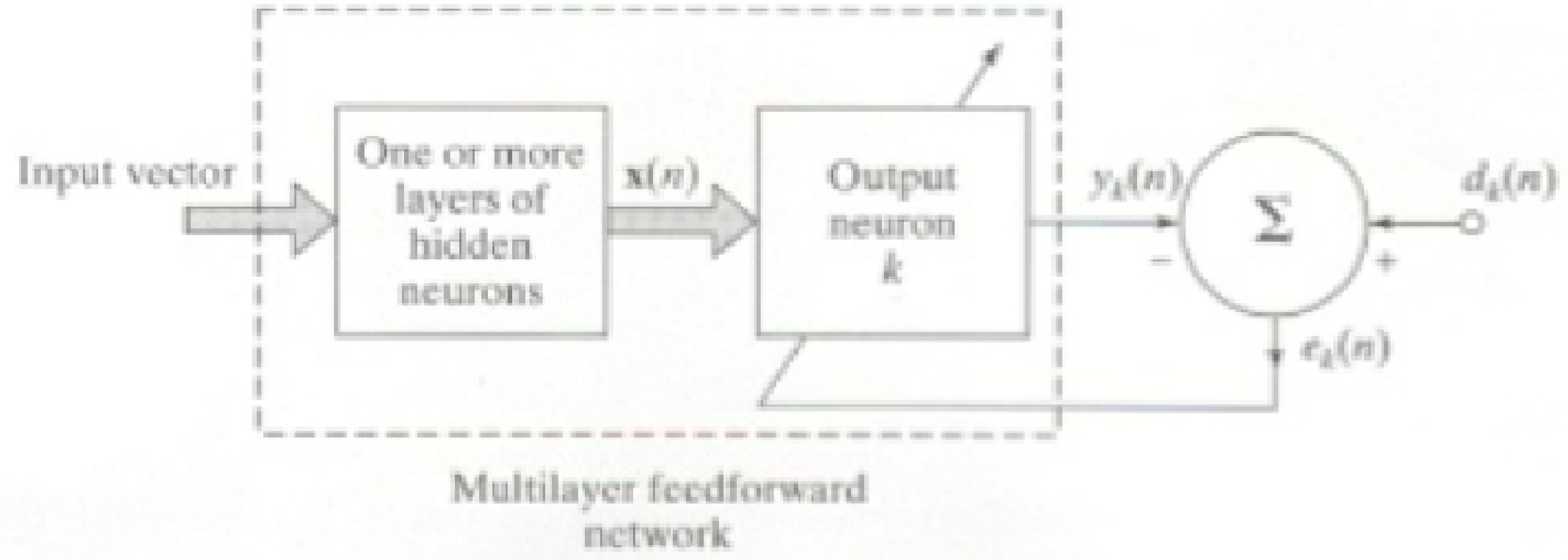
Fig. 2: Block Diagram of an Unsupervised Training Model

Learning Rules:

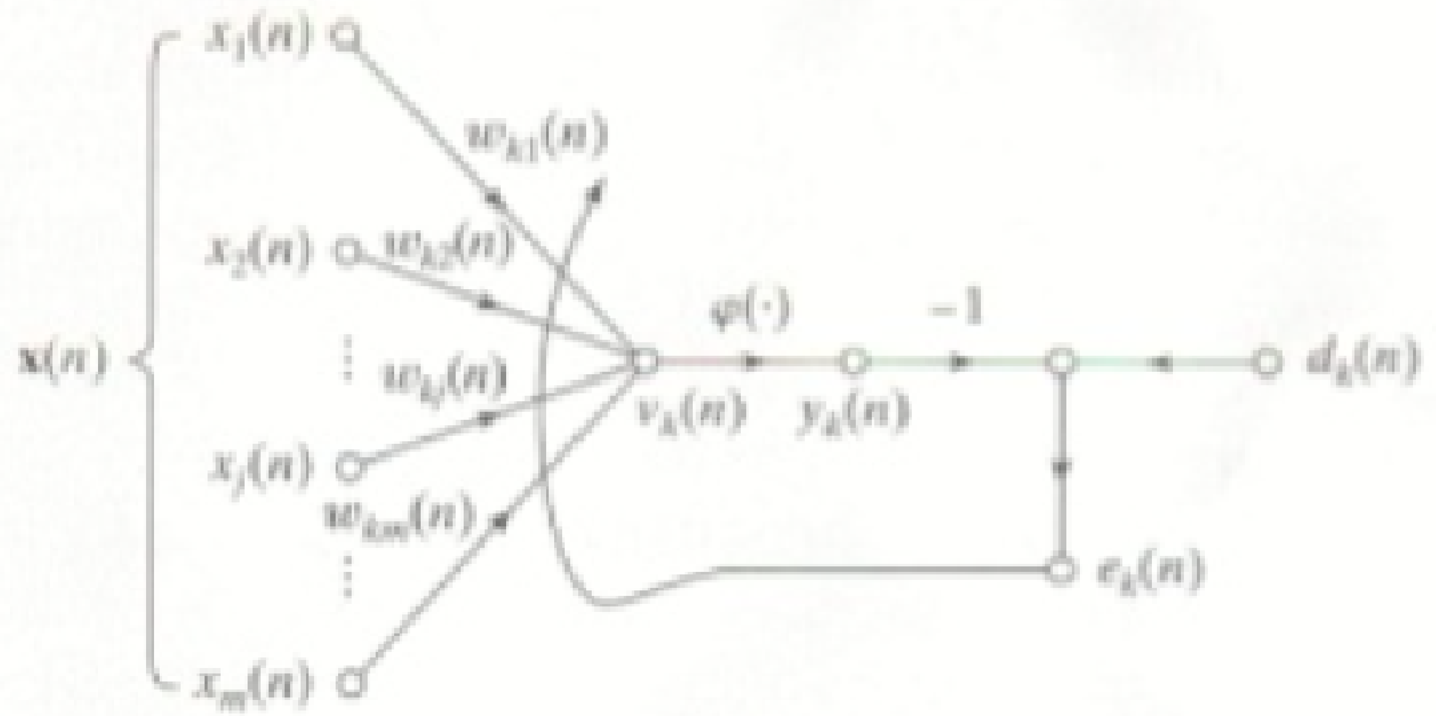


- Error correction learning
- Memory based learning
- Hebbian learning
- Competitive learning
- Boltzmann learning

Error-correction learning:

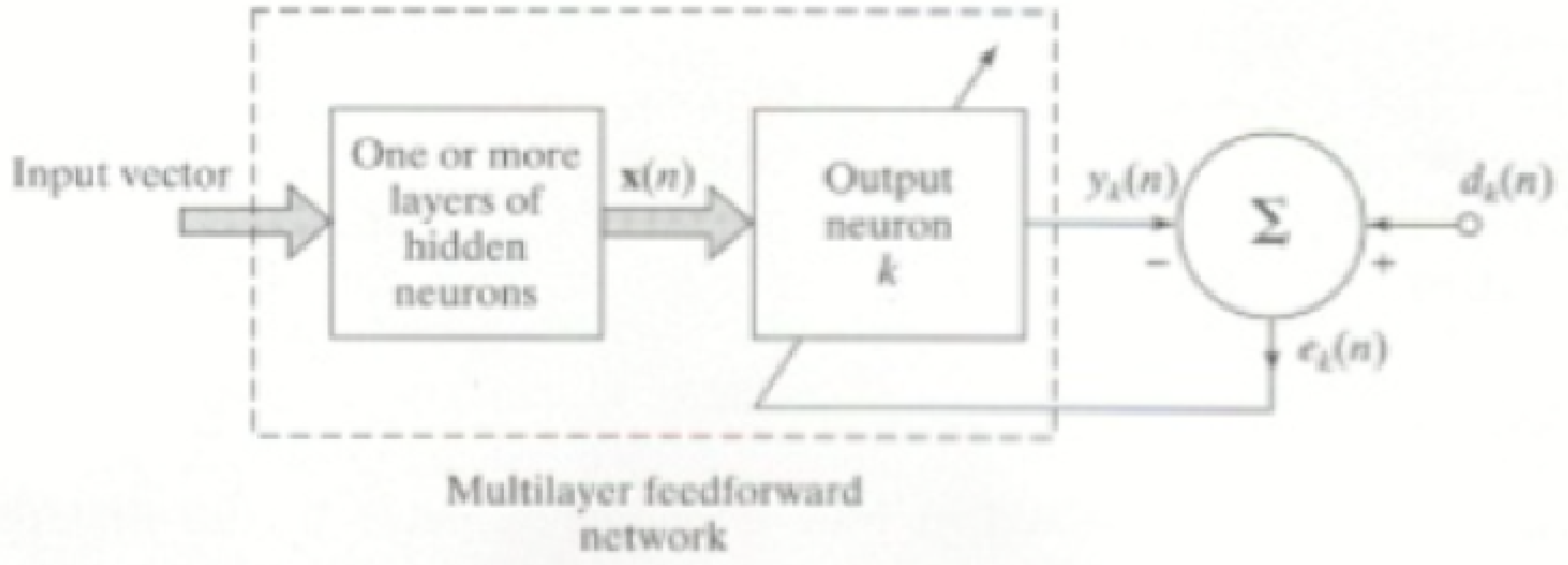


(a) Block diagram of a neural network, highlighting the only neuron in the output layer

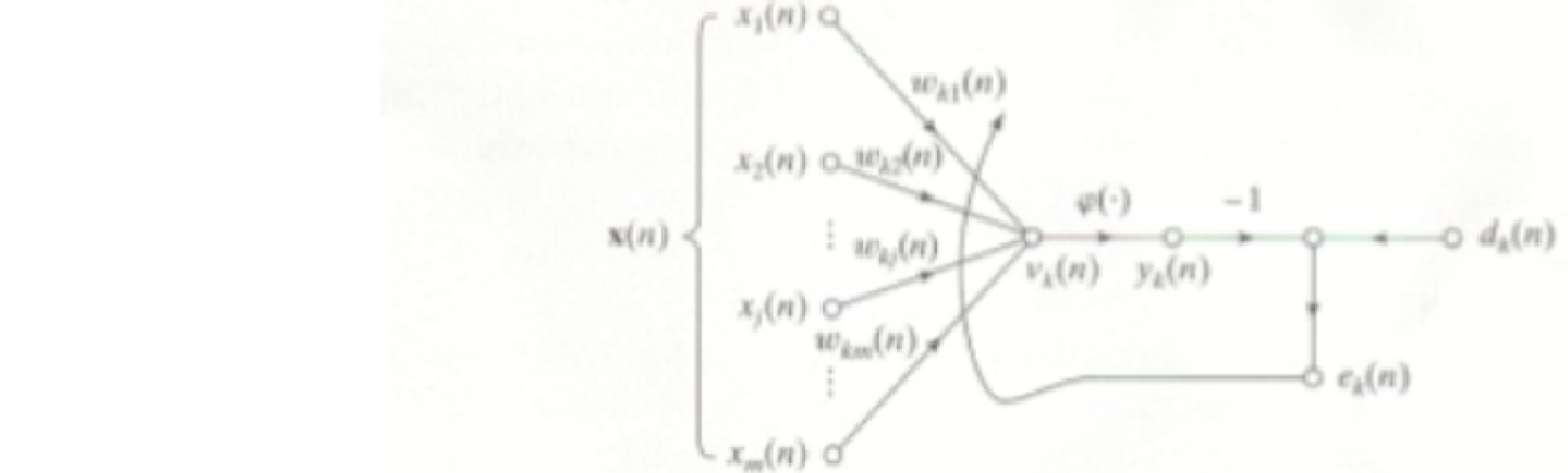


(b) Signal-flow graph of output neuron

ERROR-CORRECTION LEARNING:



(a) Block diagram of a neural network, highlighting the only neuron in the output layer



(b) Signal-flow graph of output neuron

Neuron K is driven by a signal vector x(n) produced by one more layers of hidden neurons.Which are themselves driven by an input vector applied to the source nodes

→ The O/P signal of neuron K is $y_K(n)$