Machine Learning
CSDLO6021

Subject Incharge

Anuradha Srinivasaraghavan
Associate Professor
Room No. 401
email: g.anuradha@sfit.ac.in

Module II

Introduction to Neural Networks

Learning Objectives

- Introduction-Fundamental Concept
- Evolution of Neural Networks
- Biological Neuron, Artificial Neural Networks
- Different types of connections of NN, Learning and activation function
- Basic fundamental neuron model-McCulloch-Pitts neuron and Hebb network

Introduction

- Humans are good in narration whereas computers are good in computations
- The art of story telling which humans have is not possessed by a computer.
- AI tries to accommodate the capabilities of both computers and humans.

Introduction to ANN

- Tries to mimic the structure and function of our nervous system
- Used as a methodology for information processing and the method got its inspiration from biological nervous systems.
- These system consists of highly inter connected neurons working together to solve different kinds of problems.

Use of ANN

- NN is capable of deriving information from complicated or imprecise data
- Trends can be extracted which are too complex for humans to performs
- A trained NN can behave as an expert system

Uses of ANN contd….

- Adaptive Learning-Learning from experience
- Self-Organization-ANN is capable of organizing and representing the information it receives from training data.
- Real-Time Operation: Parallel computations
- Fault Tolerance-If a neuron fails the functions continue with lesser accuracy

Reasons to study neural computation

- To understand how brain actually works
  - Computer simulations are used for this purpose
- To understand the style of parallel computation inspired by neurons and their adaptive connections
  - Different from sequential computation
- To solve practical problems by using novel learning algorithms inspired by brain

Evolution of Neural Networks

Reticular Theory

Joseph von Gerlach proposed that the nervous system is a single continuous network as opposed to a network of many discrete cells!

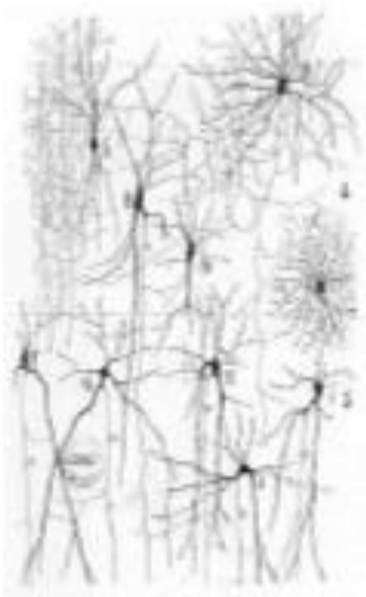
Staining Technique

Camillo Golgi discovered a chemical reaction that allowed him to examine nervous tissue in much greater detail than ever before

He was a proponent of Reticular theory.

Neuron Doctrine

Santiago Ramón y Cajal used Golgi's technique to study the nervous system and proposed that it is actually made up of discrete individual cells formimg a network (as opposed to a single continuous network)



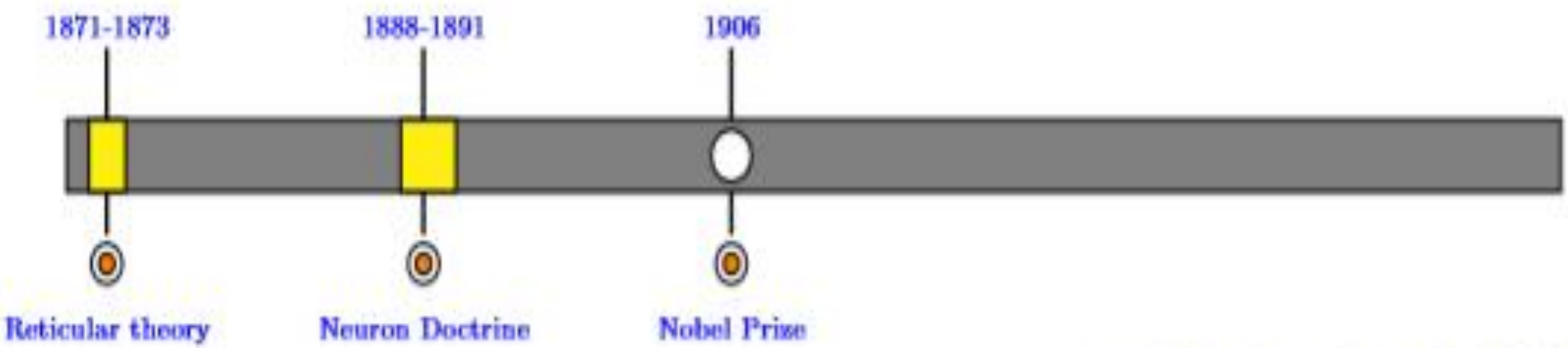
The Term Neuron

The term neuron was coined by Heinrich Wilhelm Gottfried von Waldeyer-Hartz around 1891.

He further consolidated the Neuron Doctrine.

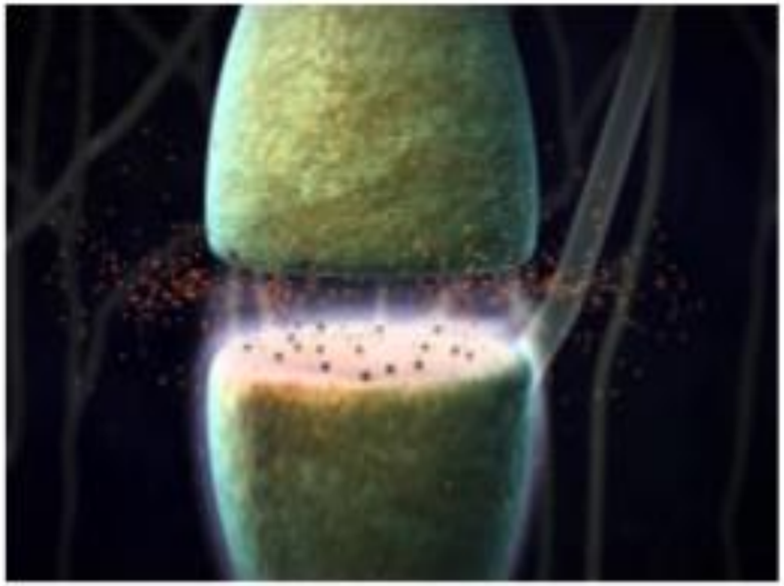
Nobel Prize

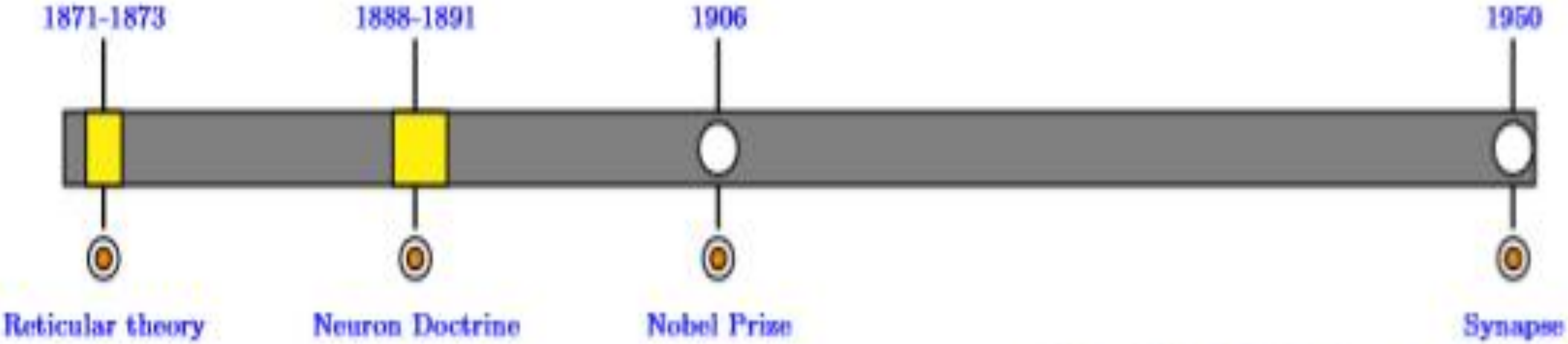
Both Golgi (reticular theory) and Cajal (neuron doctrine) were jointly awarded the 1906 Nobel Prize for Physiology or Medicine, that resulted in lasting conflicting ideas and controversies between the two scientists.



The Final Word

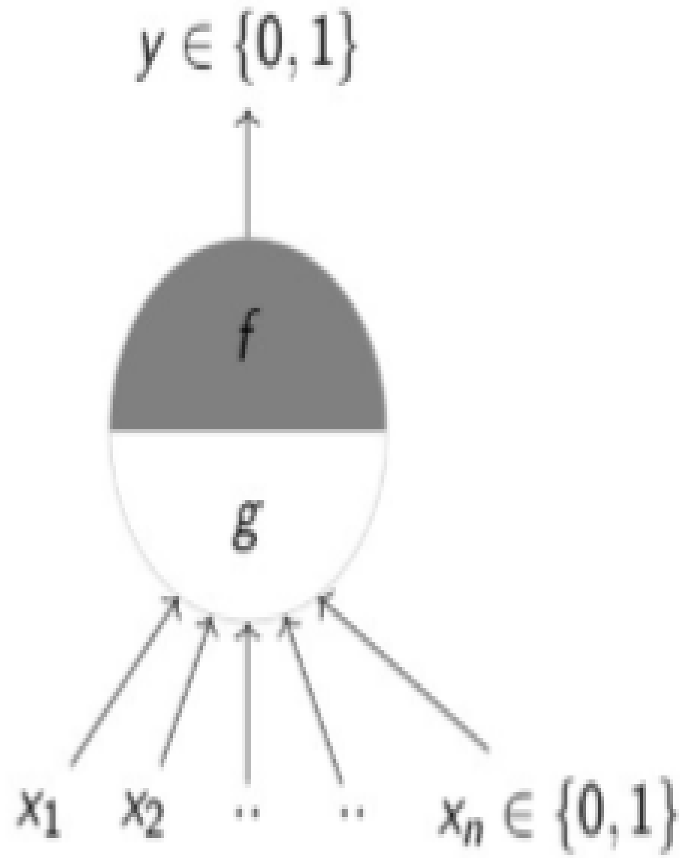
In 1950s electron microscopy finally confirmed the neuron doctrine by unambiguously demonstrated that nerve cells were individual cells interconnected through synapses (a network of many individual neurons).





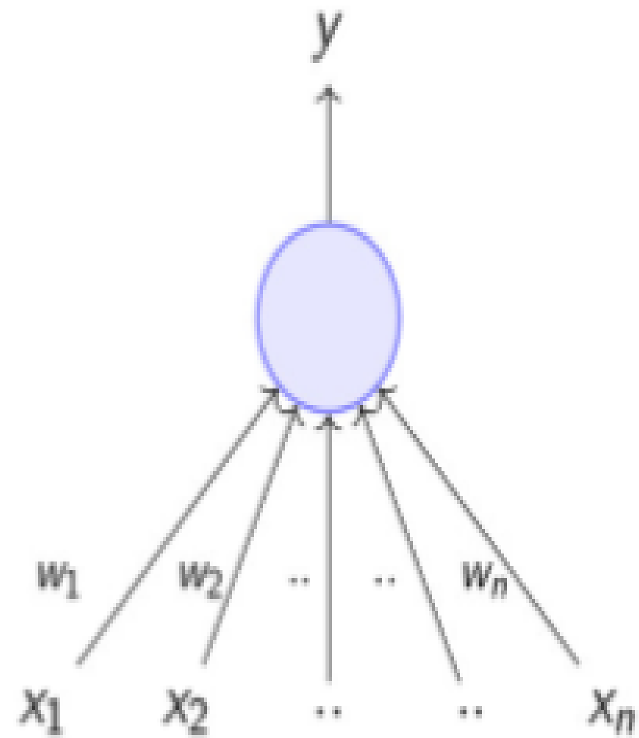
McCulloch Pitts Neuron

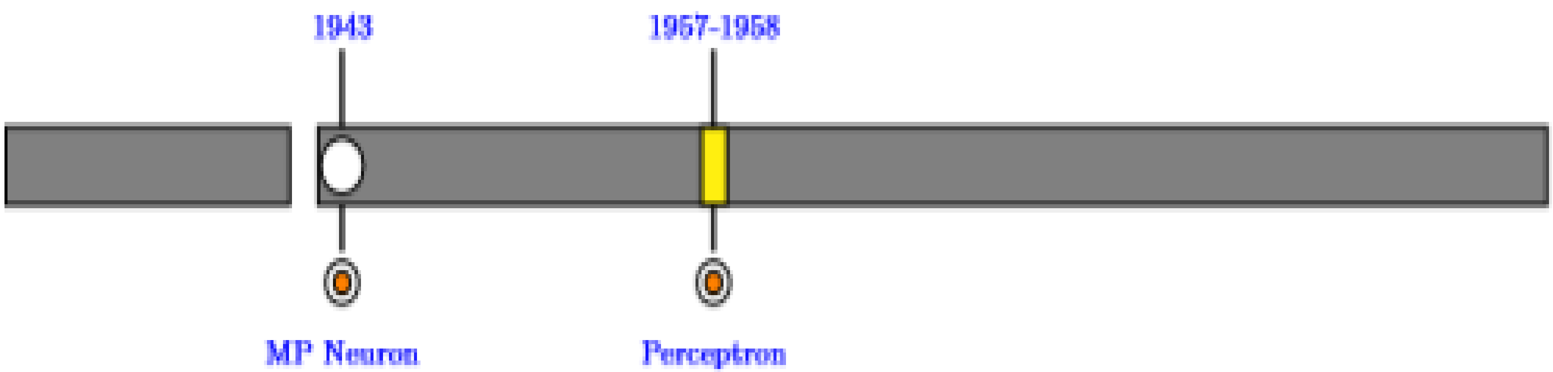
McCulloch (neuroscientist) and Pitts (logician) proposed a highly simplified model of the neuron (1943)[2]



Perceptron

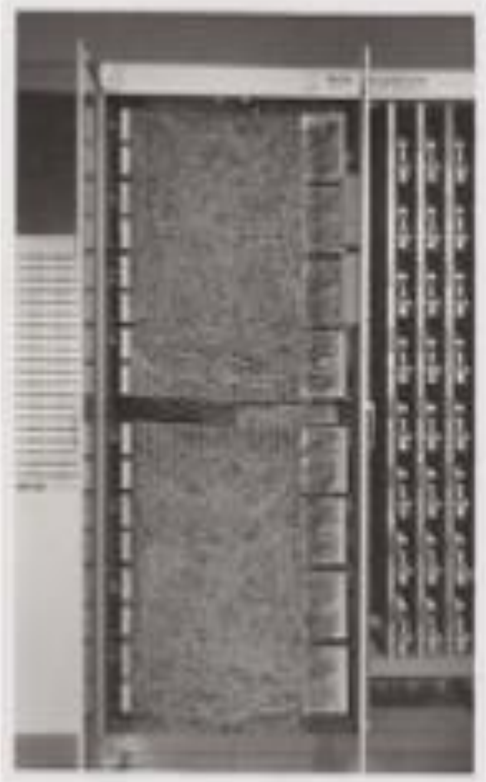
"the perceptron may eventually be able to learn, make decisions, and translate languages" -Frank Rosenblatt

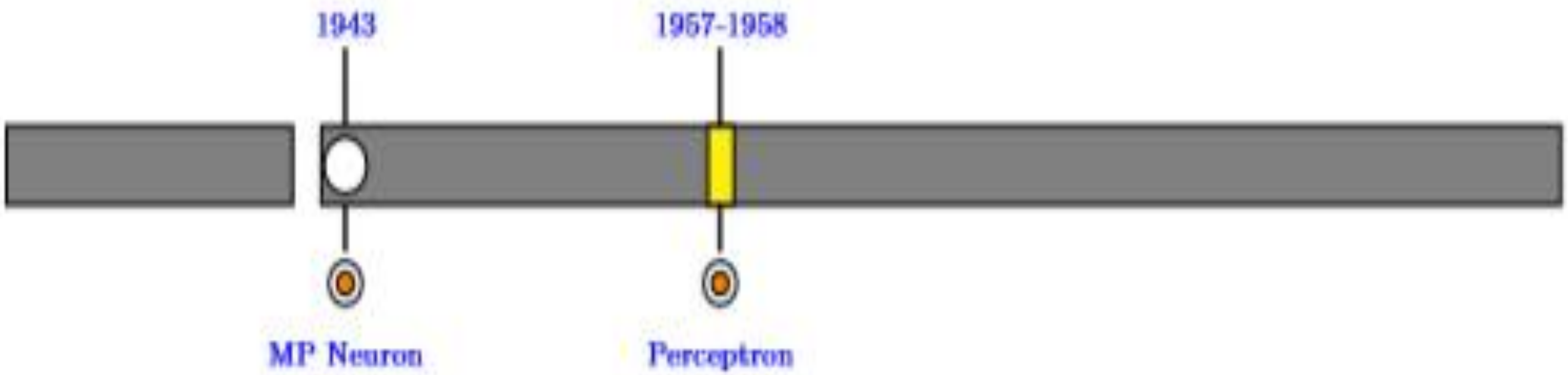




Perceptron

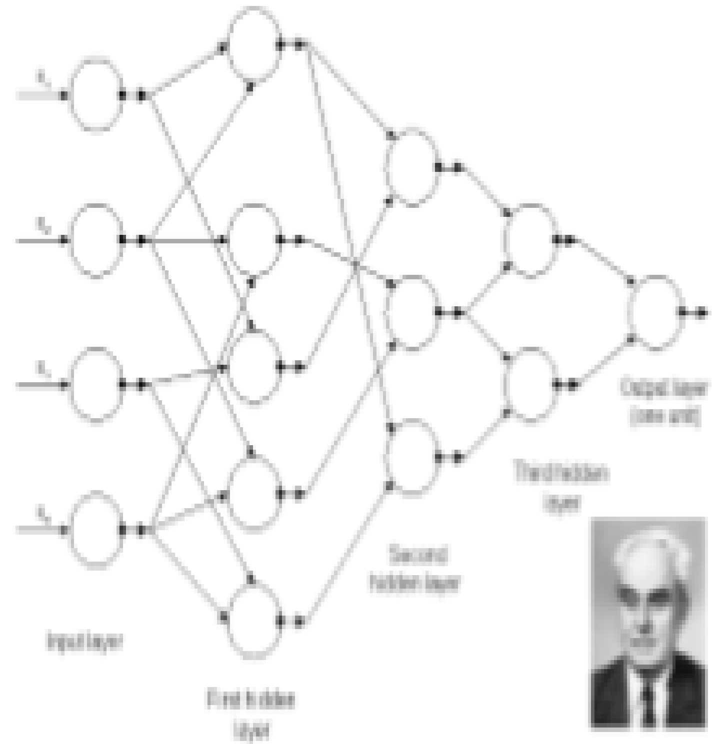
"the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence." -New York Times

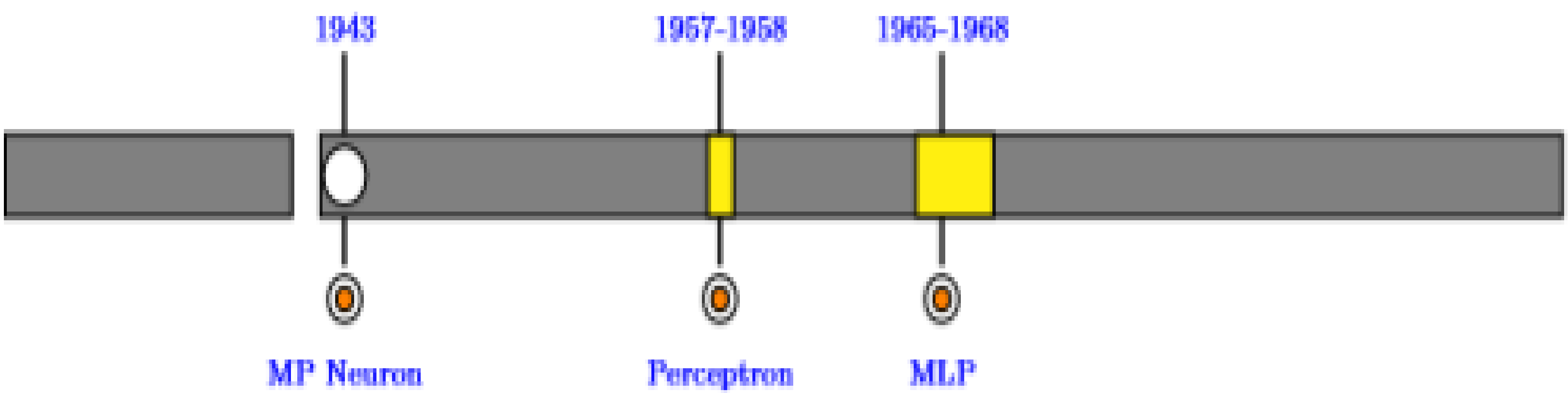




First generation Multilayer Perceptrons

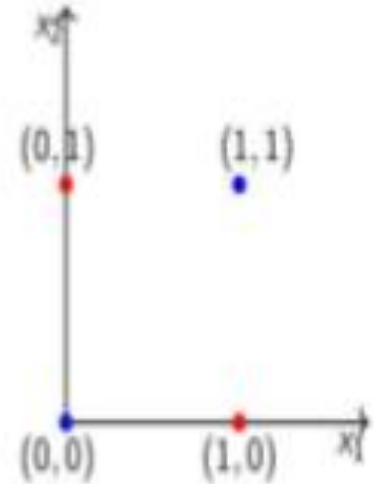
Ivakhnenko et. al.[3]

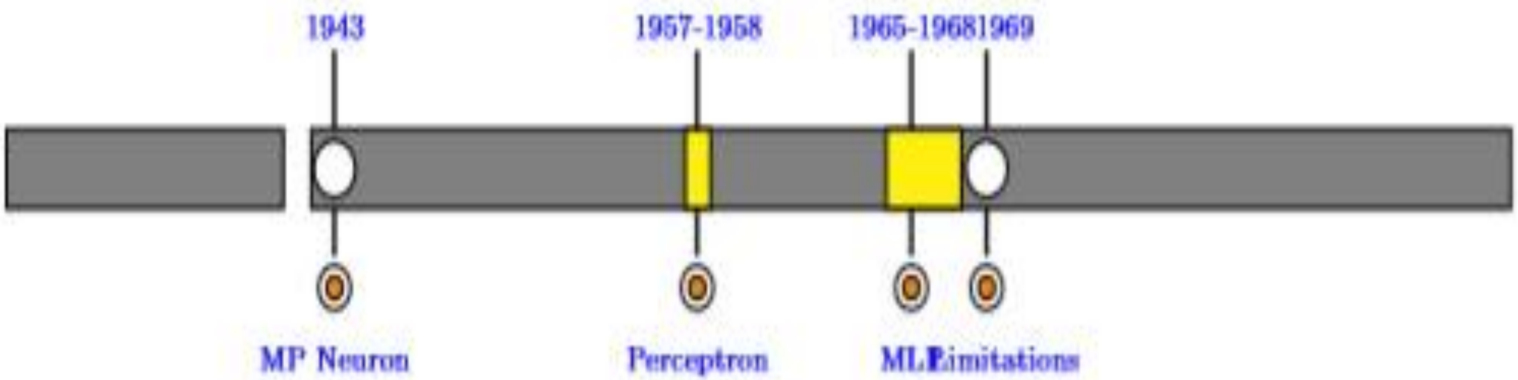




Perceptron Limitations

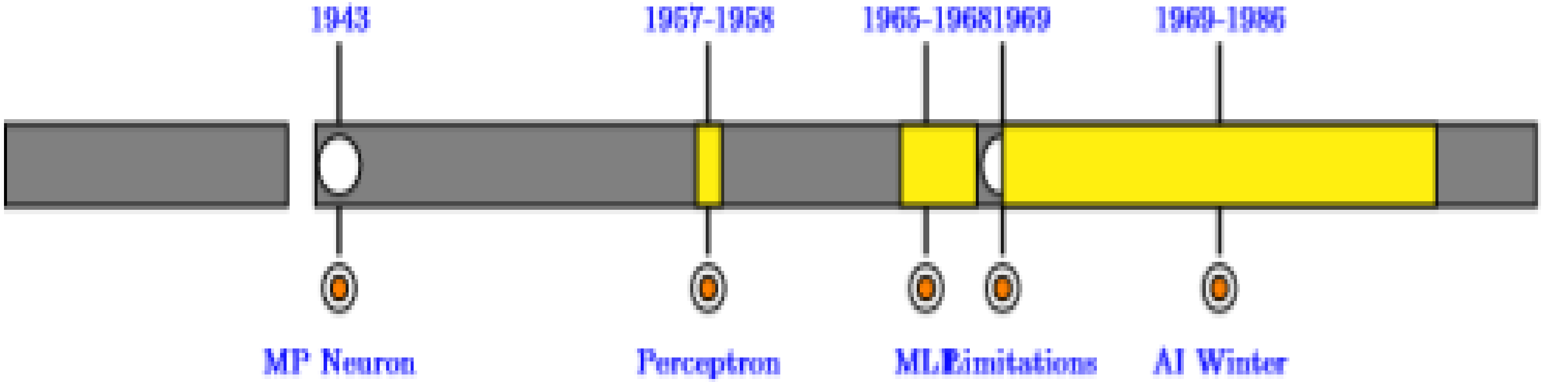
In their now famous book "Perceptrons", Minsky and Papert outlined the limits of what perceptrons could do[4]





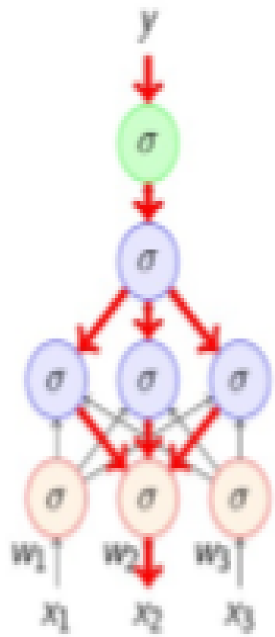
AI Winter of connectionism

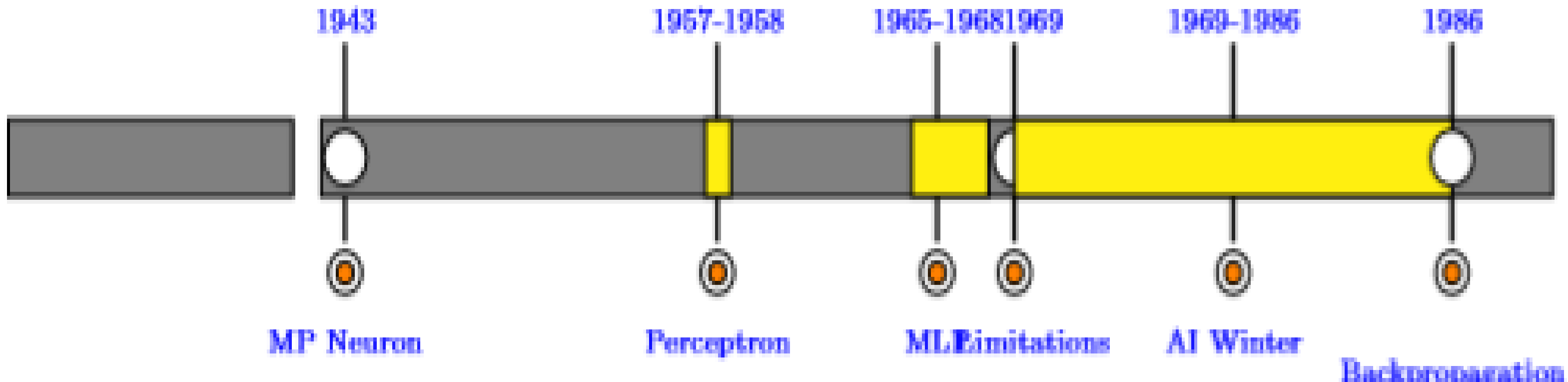
Almost lead to the abandonment of connectionist AI



Backpropagation

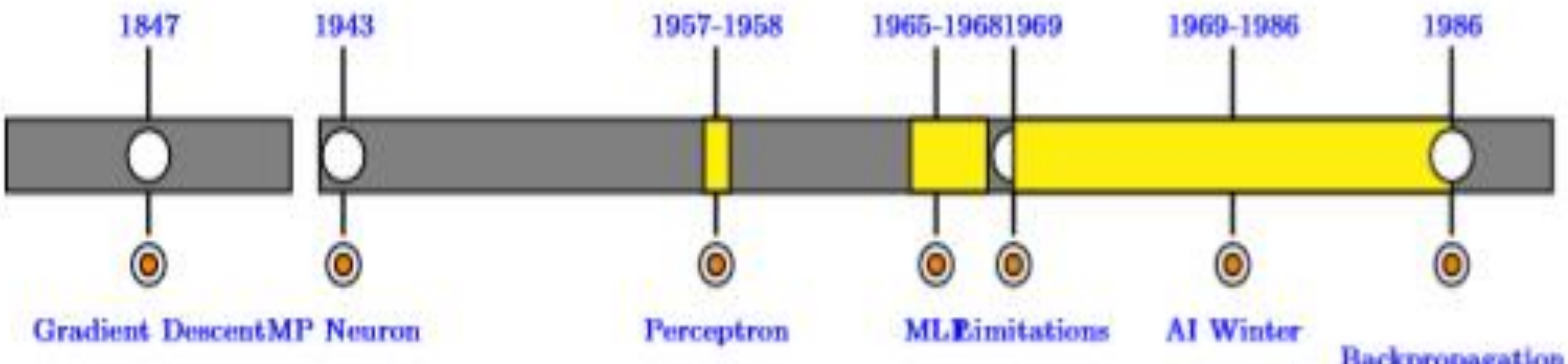
- Discovered and rediscovered several times throughout 1960's and 1970's
- Werbos(1982)[5] first used it in the context of artificial neural networks
- Eventually popularized by the work of Rumelhart et. al. in 1986[6]





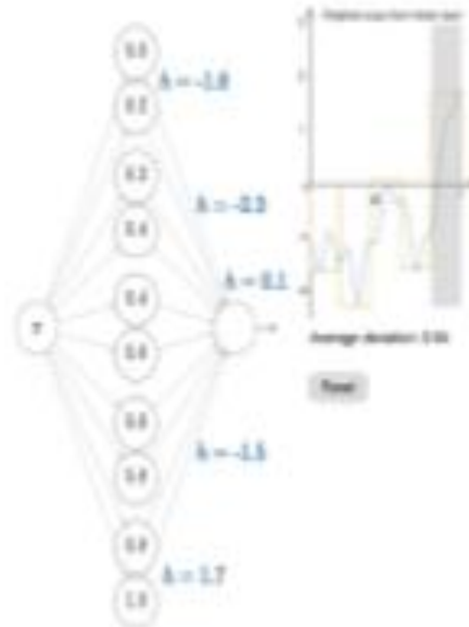
Gradient Descent

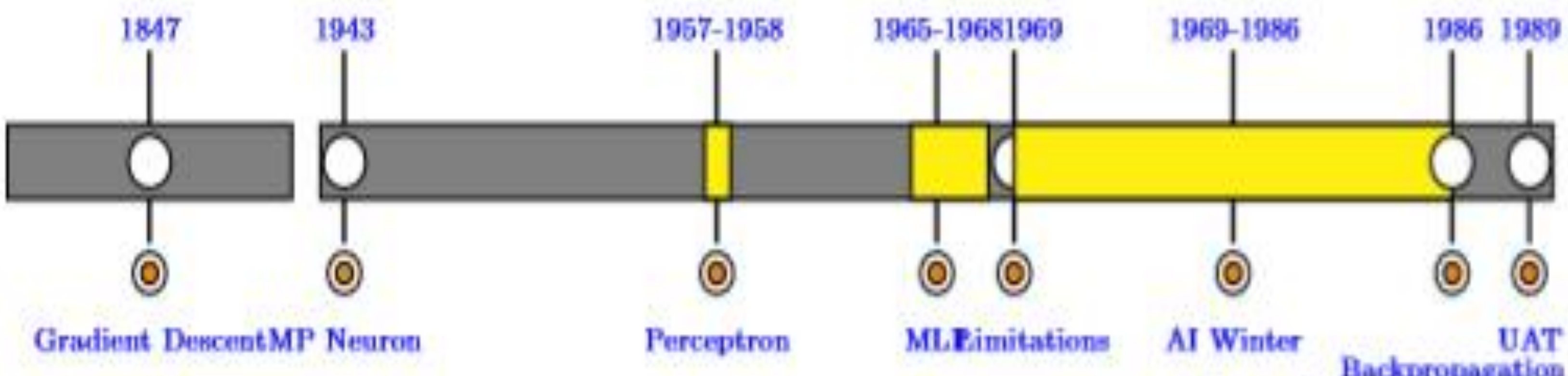
Cauchy discovered Gradient Descent motivated by the need to compute the orbit of heavenly bodies



Universal Approximation Theorem

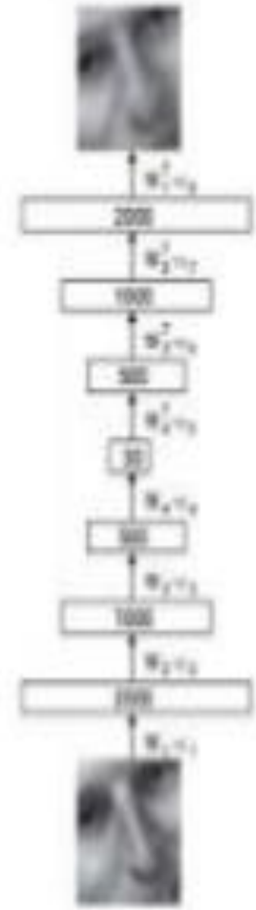
A multilayered network of neurons with a single hidden layer can be used to approximate any continuous function to any desired precision[7]





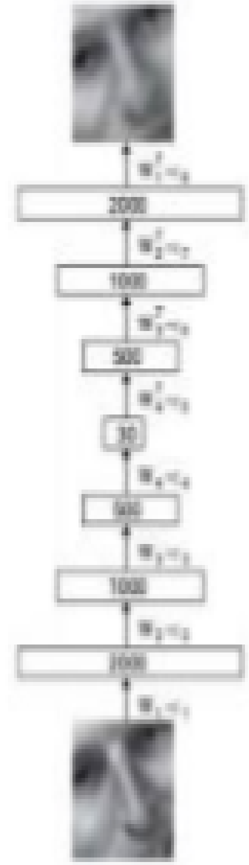
Unsupervised Pre-Training

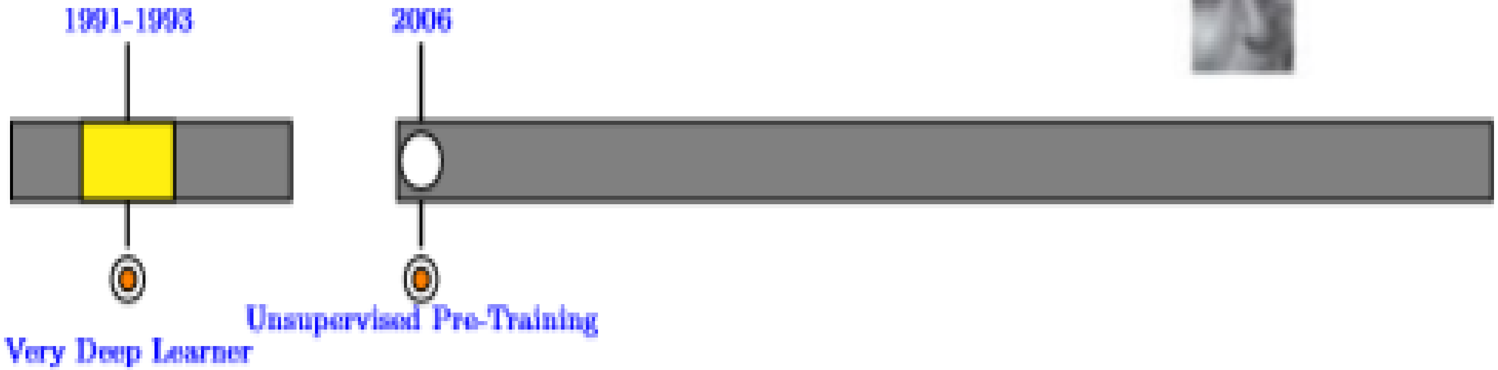
Hinton and Salakhutdinov described an effective way of initializing the weights that allows deep autoencoder networks to learn a low-dimensional representation of data.[8]



Unsupervised Pre-Training

The idea of unsupervised pre-training actually dates back to 1991-1993 (J. Schmidhuber) when it was used to train a "Very Deep Learner"





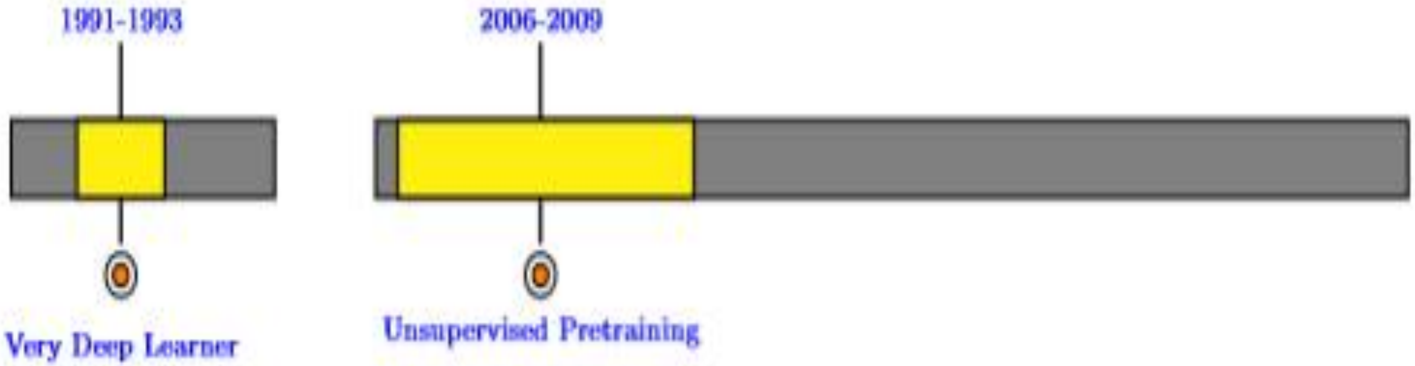
More insights (2007-2009)

Further Investigations into the effectiveness of Unsupervised Pre-training

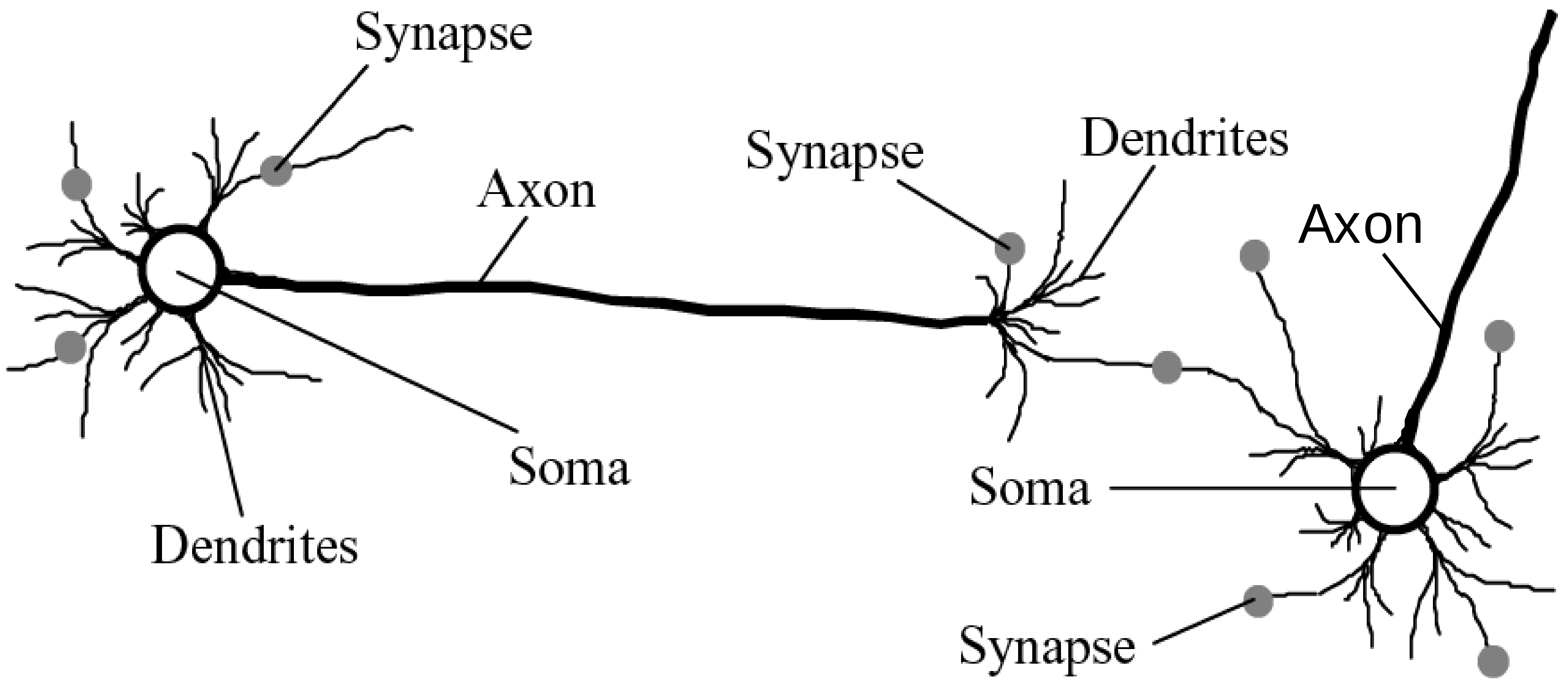
Greedy Layer-Wise Training of Deep Networks

Why Does Unsupervised Pre-training Help Deep Learning?

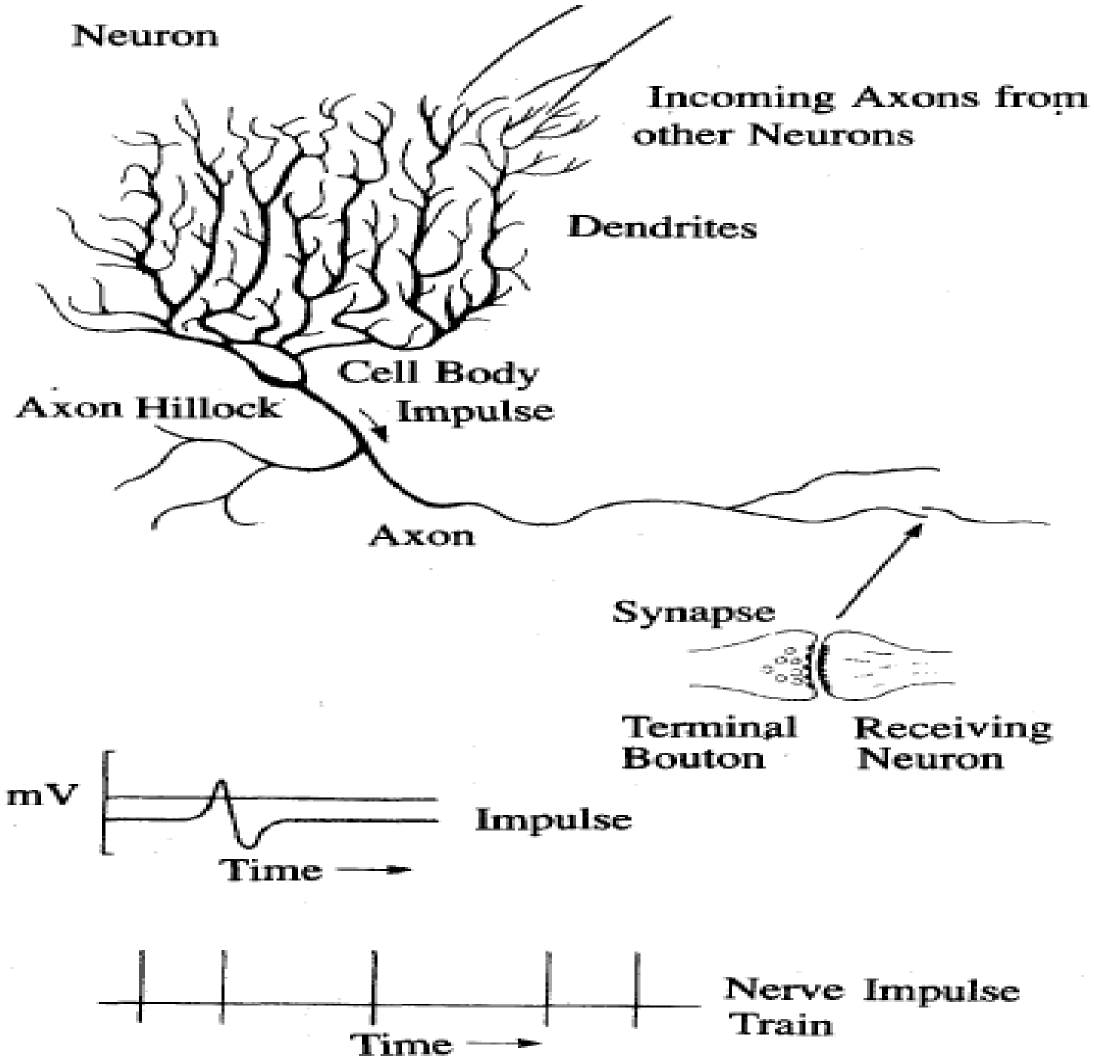
Exploring Strategies for Training Deep Neural Networks



Biological Neural Network



Neuron and a sample of pulse train



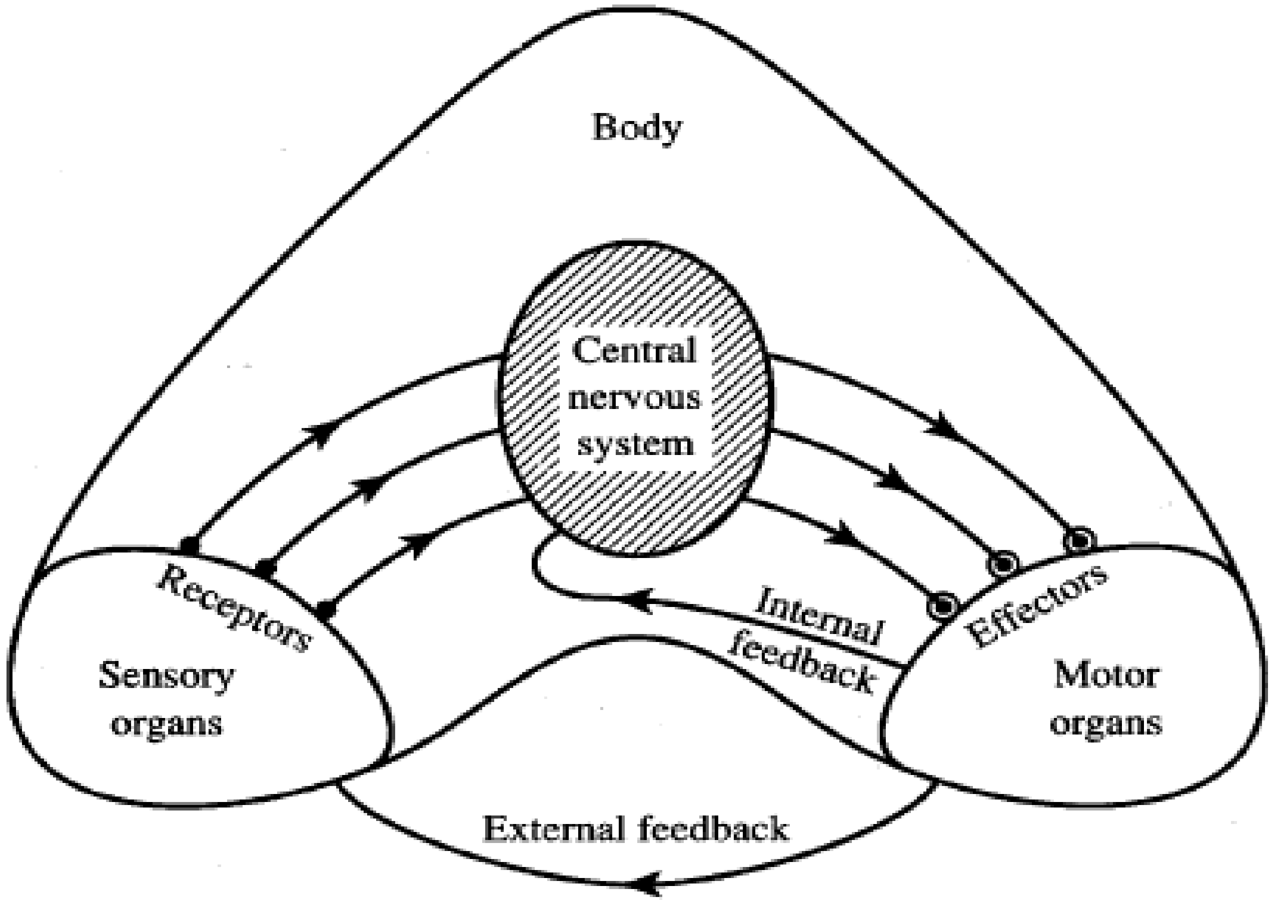
How does the brain work

- Each neuron receives inputs from other neurons
  - Use spikes to communicate
- The effect of each input line on the neuron is controlled by a synaptic weight
  - Positive or negative
- Synaptic weight adapts so that the whole network learns to perform useful computations
  - Recognizing objects, understanding languages, making plans, controlling the body
- There are 10^{11} neurons with 10^{4} weights. By using multiple neurons simultaneously, the brain can perform its functions much faster than the fastest computers in existence today.

Contd..

- Our brain can be considered as a highly complex, non-linear and parallel information-processing system.
- Learning is a fundamental and essential characteristic of biological neural networks.

Information flow in nervous system



ANN

- Artificial neural network (ANN) is a machine learning approach that models human brain and consists of a number of artificial neurons.
- ANN possess a large number of processing elements called nodes/neurons which operate in parallel.
- Neuron in ANNs tend to have fewer connections than biological neurons.
- Neurons are connected with others by connection link.
- Each link is associated with weights which contain information about the input signal.
- Each neuron has an internal state of its own which is a function of the inputs that neuron receives- <u>Activation level</u>

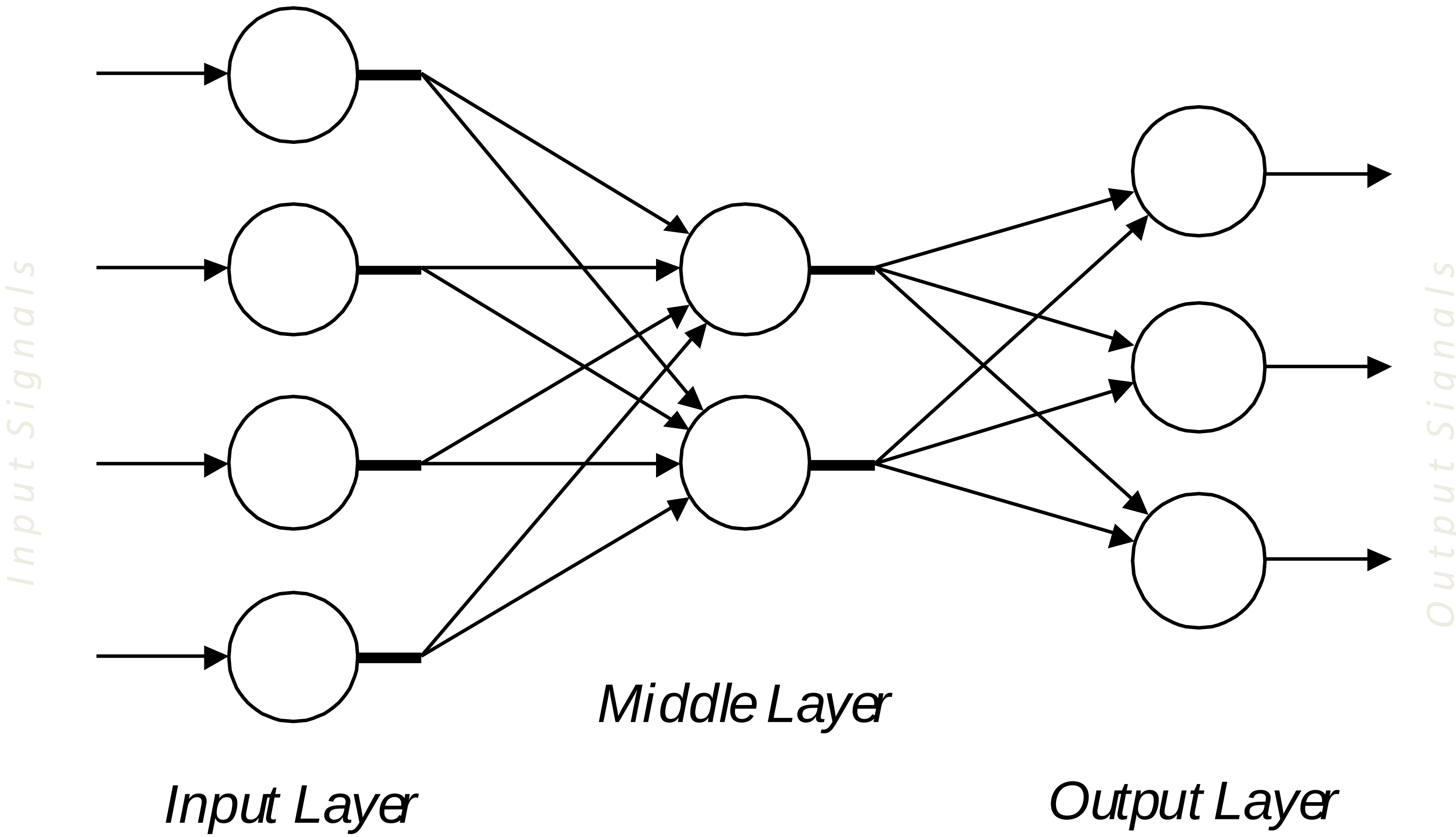
Contd..

- Each neuron in ANN receives a number of inputs.
- An activation function is applied to these inputs which results in activation level of neuron (output value of the neuron).
- Knowledge about the learning task is given in the form of examples called training examples.

Contd..

- An Artificial Neural Network is specified by:
  - **neuron model**: the information processing unit of the NN,
  - **an architecture**: a set of neurons and links connecting neurons. Each link has a weight,
  - **a learning algorithm**: used for training the NN by modifying the weights in order to model a particular learning task correctly on the training examples.
- The aim is to obtain a NN that is trained and generalizes well.
- It should behave correctly on new instances of the learning task.

Architecture of a typical artificial neural network

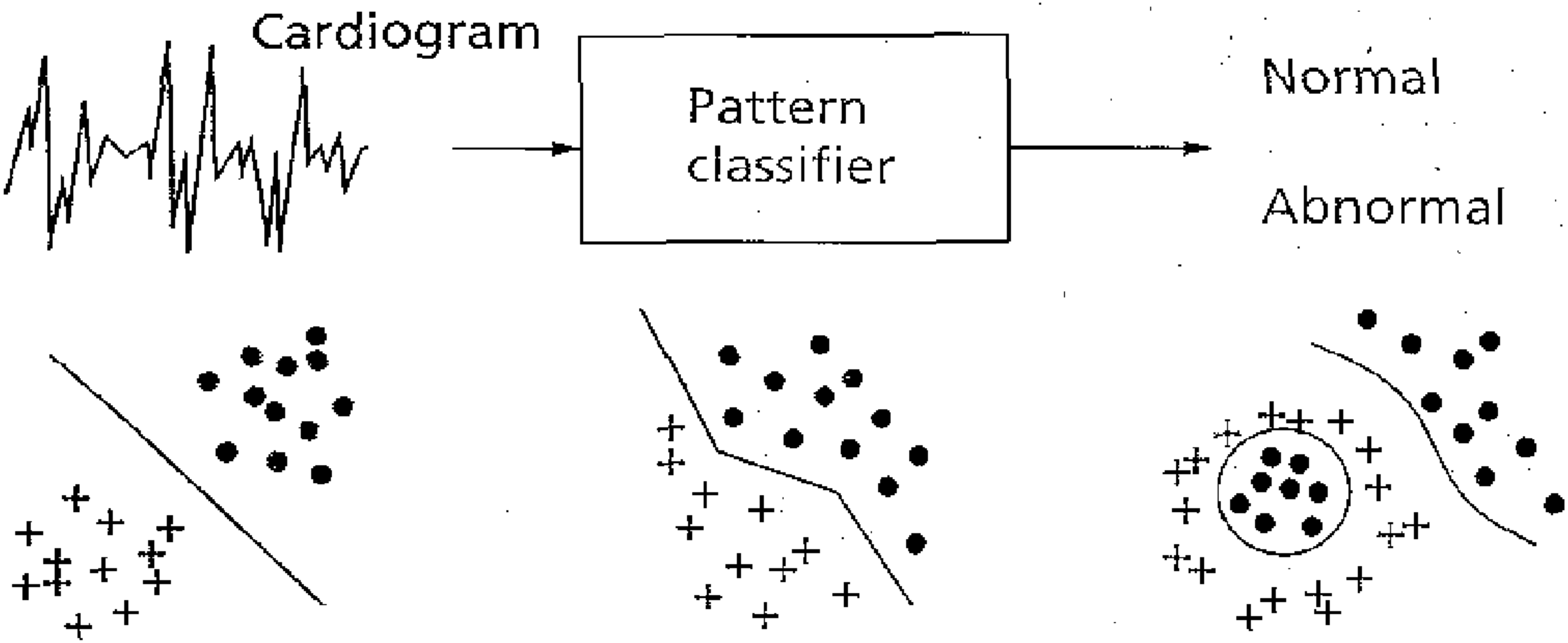


Analogy between biological and artificial neural networks

Biological Neural Network	***Artificial Neural Network***
Soma	Neuron
Dendrite	Input
Axon	Output
Synapse	Weight

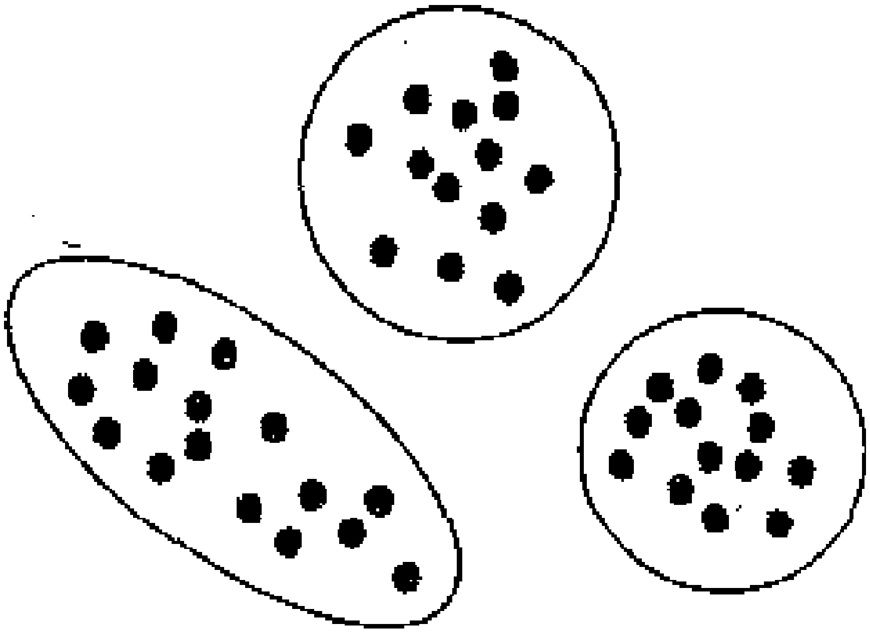
ANN Applications

- **Pattern Classification**
  - **Speech Recognition, ECG/EEG classification etc.**



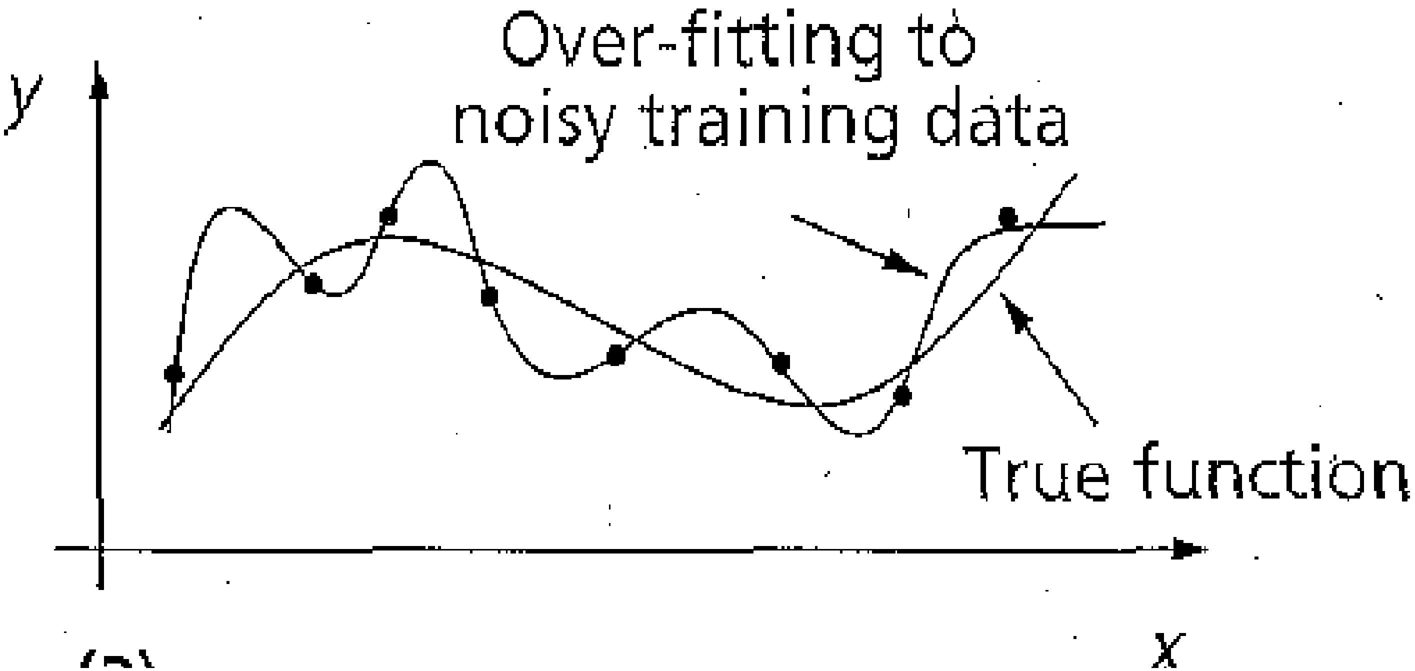
ANN Applications

- **Clustering/Categorization**
  - **Data mining, data compression**



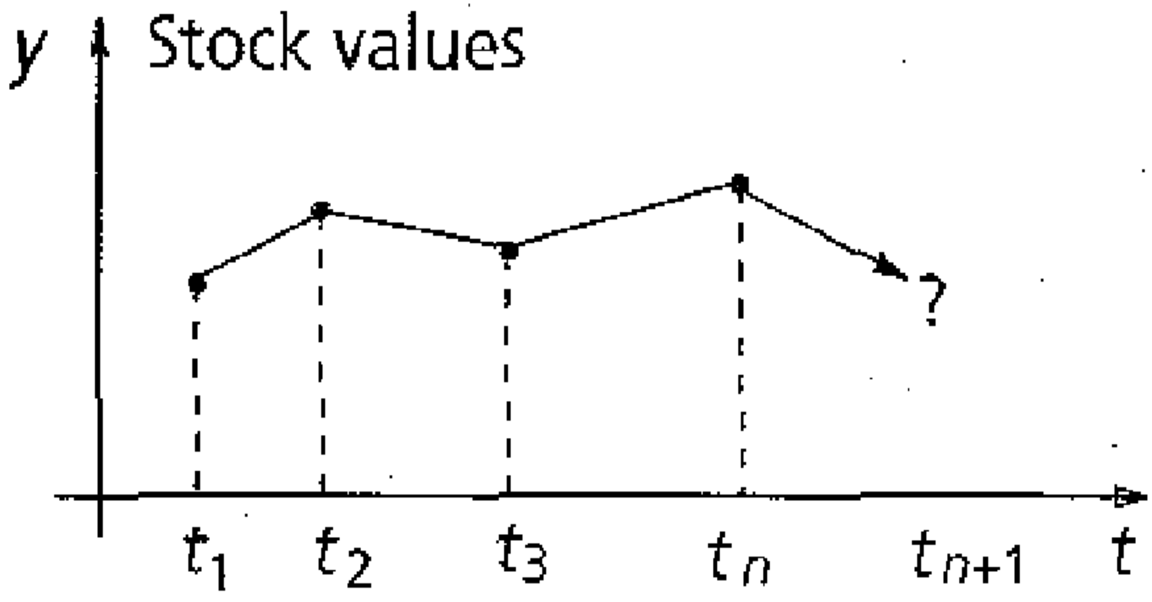
ANN Applications

- **Function Approximation**
  - **Noisy arbitrary function needs to be approximated**



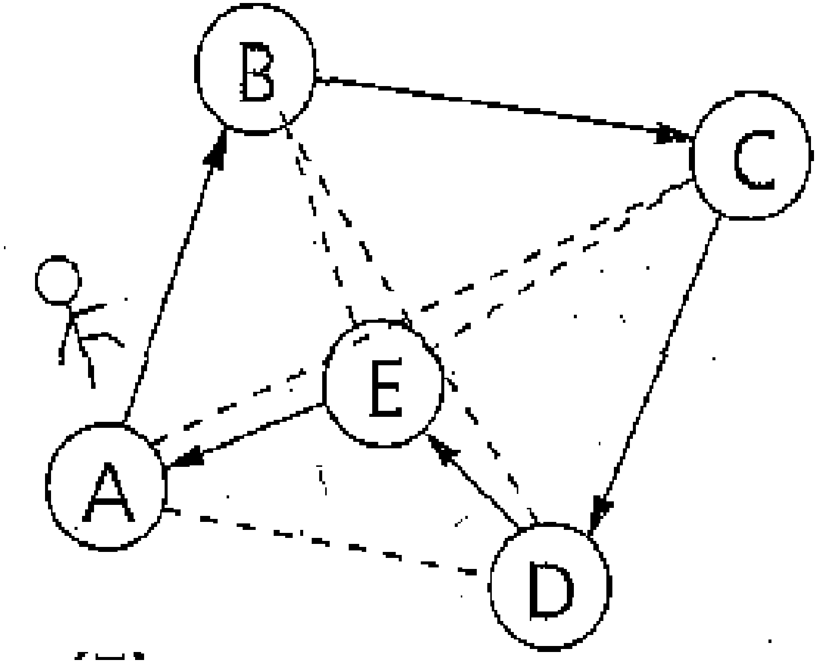
ANN Applications

- **Prediction/Forecasting**
  - **Given a function of time, predict the function values for future time values, used in weather prediction and stock market predictions**



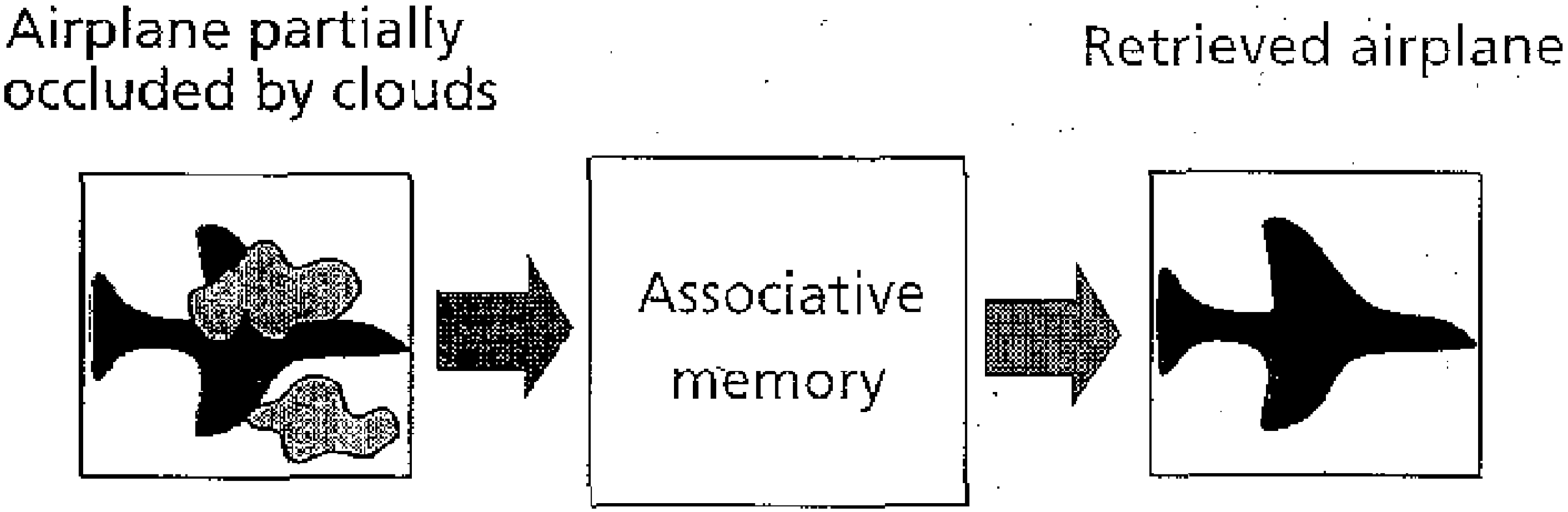
ANN Applications

- **Optimization**
  - **Several scientific and other problems can be reduced to an optimization problem like the Traveling Salesman Problem (TSP)**



ANN Applications

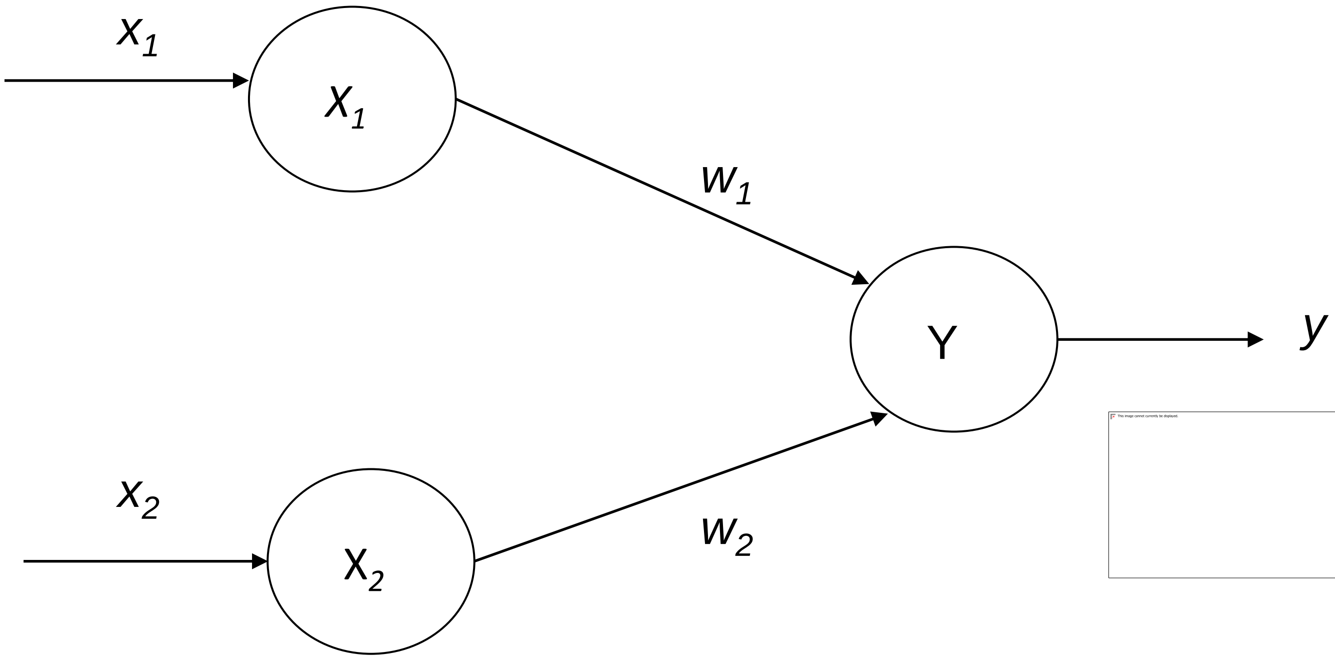
- Content Based Retrieval
  - Given the partial description of an object retrieve the objects that match this

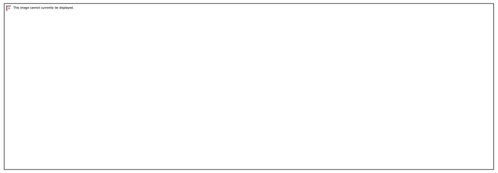


Comparison between brain verses computer

	Brain	ANN
Speed	Few ms.	Few nano sec. massive ||el processing
Size and complexity	10^{11} neurons & 10^{15} interconnections	Depends on designer
Storage capacity	Stores information in its interconnection or in synapse. No Loss of memory	Contiguous memory locations loss of memory may happen sometimes.
Tolerance	Has fault tolerance	No fault tolerance Inf gets disrupted when interconnections are disconnected
Control mechanism	Complicated involves chemicals in biological neuron	Simpler in ANN

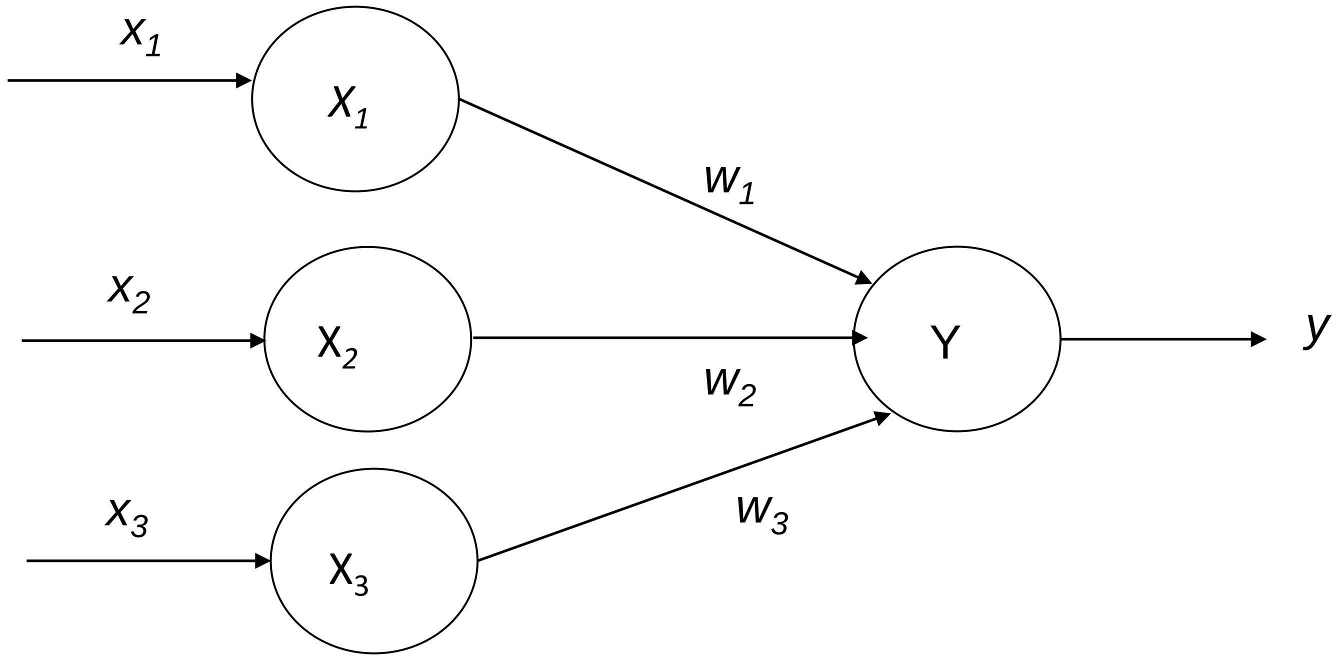
Artificial Neural Networks





$$y_{in} = x_1 w_1 + x_2 w_2$$

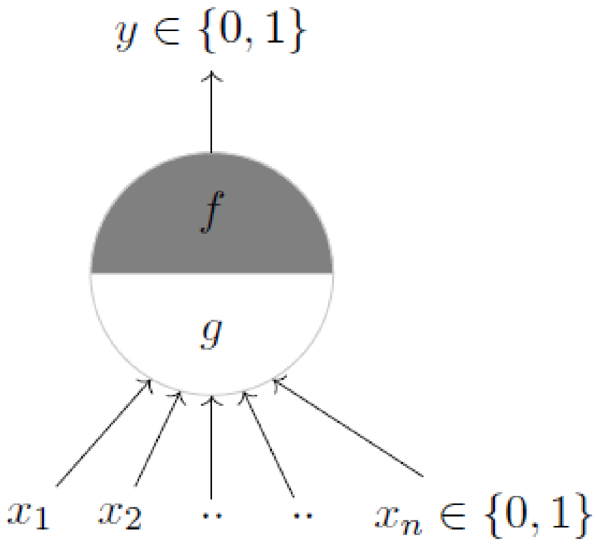
Numerical



Compute Yin given X=[2,3,4] and w^T=[.0,.2,-.1]

McCulloch-Pitts Neuron Model

- McCulloch (neuroscientist) and Pitts (logician) proposed a highly simplified computational model of the neuron (1943)
- ***g*** aggregates the inputs
- ***f*** takes a decision based on this aggregation
- The inputs can be excitatory or inhibitory



- ***y = 0*** if any xi is inhibitory, else

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

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

- ***θ*** is called the thresholding parameter
- This is called Thresholding Logic

McCulloch-Pitts Neuron Model

Fixed weights:

wi= +1 for excitatory synapses,

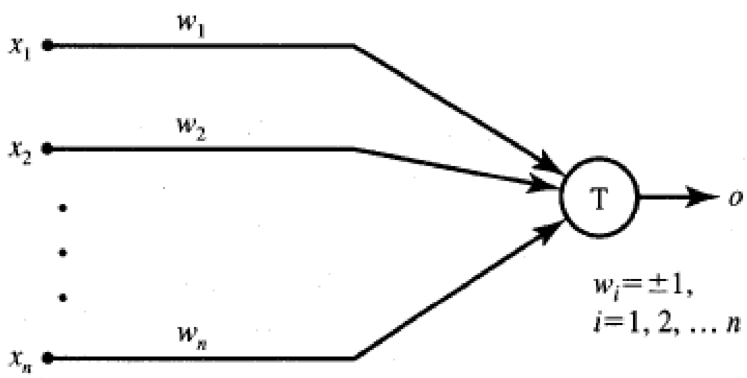
wi= -1 for inhibitory synapses

- Fixed thresholds: T is the neuron's threshold value

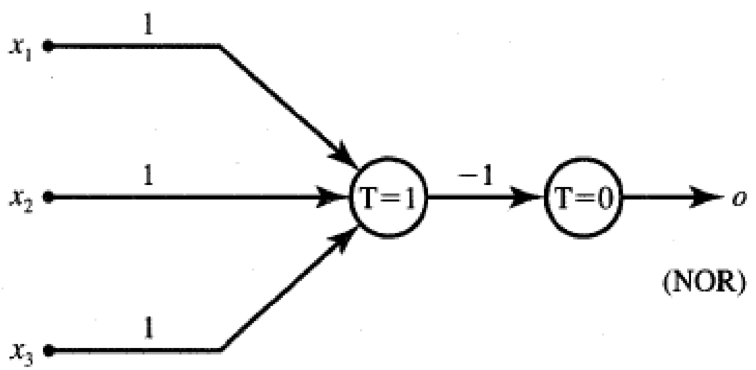
– Needs to be exceeded by the weighted sum of signals for the neuron to fire.

McCulloch-Pitts Neuron Model

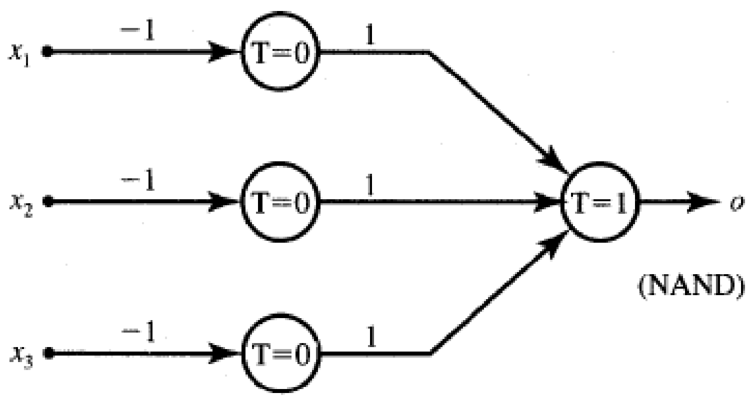
$$o^{k+1} = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i^k \geq T \\ 0 & \text{if } \sum_{i=1}^{n} w_i x_i^k < T \end{cases}$$



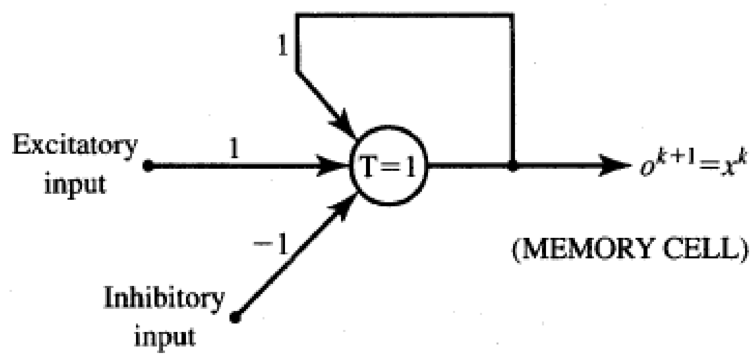
(a)



(b)



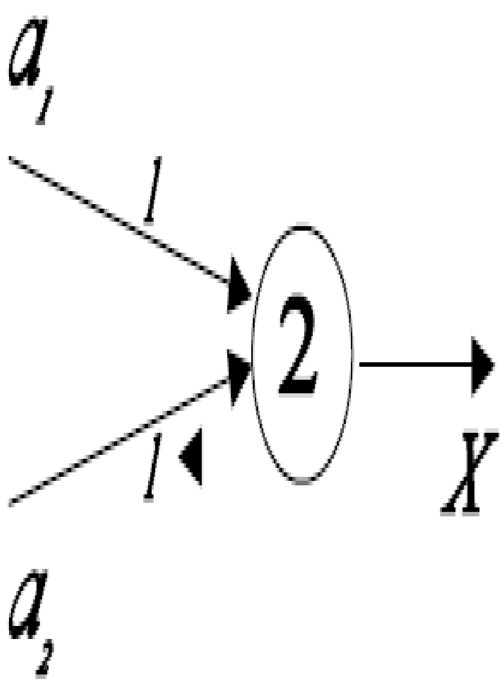
(c)



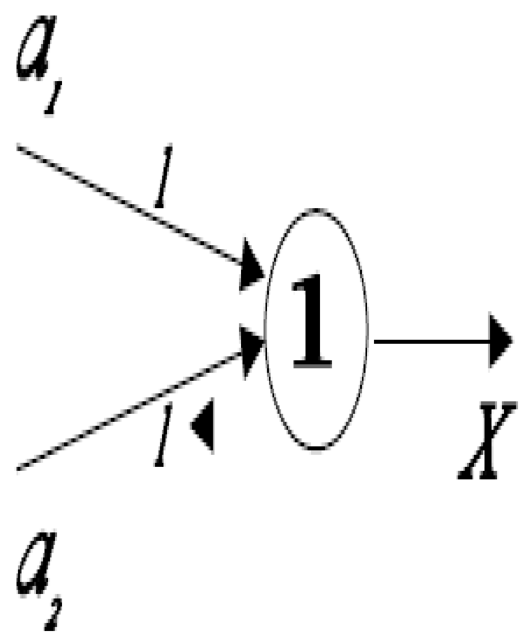
(d)

McCulloch Pits for And and or model

1) "AND" (the output fires if a_1 and a_2 both fire):

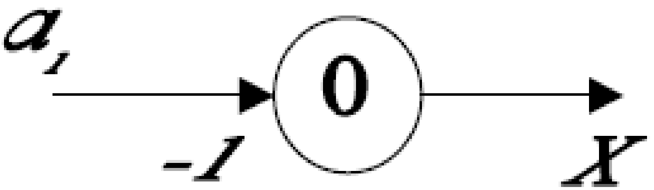


2) "OR" (the output fires if a_1 or a_2 or both fire):



McCulloch Pitts for NOT Model

3) **"NOT"** (the output fires if a_1 does NOT fire):



Advantages and Disadvantages of McCulloch Pitt model

- Advantages
- Simplistic
- Substantial computing power

- Disadvantages
  - Weights and thresholds are fixed
  - Not very flexible

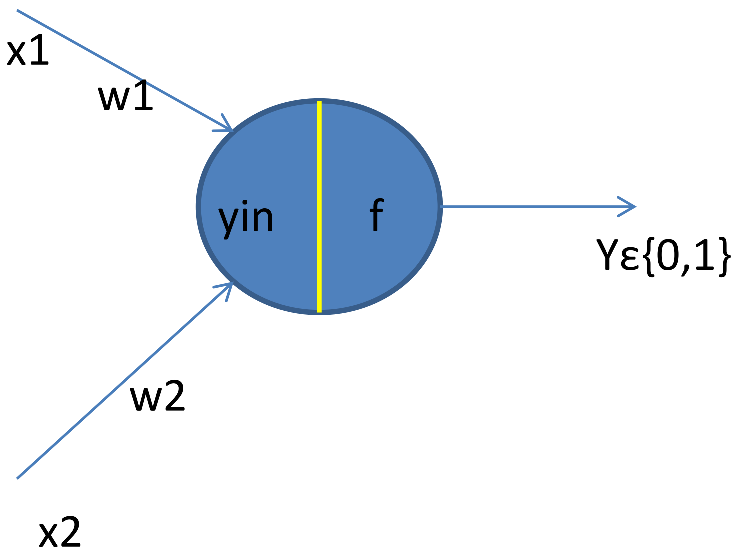
Features of McCulloch-Pitts model

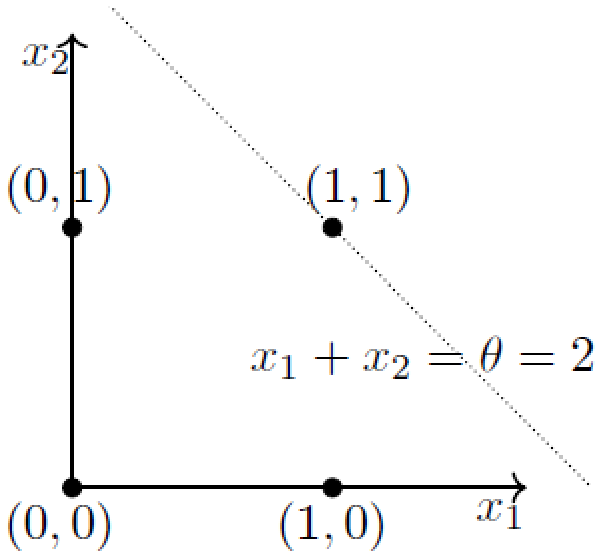
- Allows binary 0,1 states only
- Operates under a discrete-time assumption
- Weights and the neurons' thresholds are fixed in the model and no interaction among network neurons
- Just a primitive model

Implementation

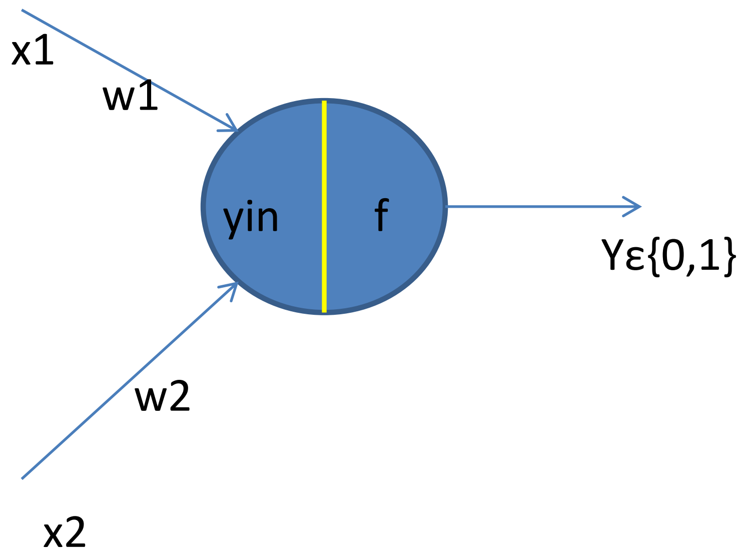
- AND
- OR
- NAND
- NOR
- ANDNOT
- XOR

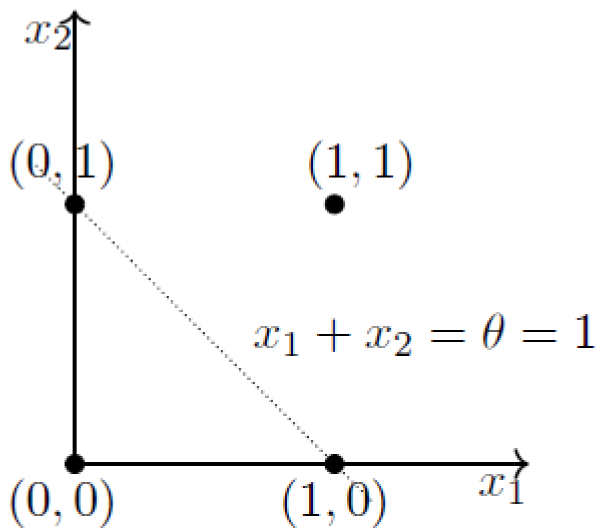
Gates	output	Inputs		Weights		yin	Threshold
AND	y	x1	x2	W1	w2		
	0	0	0	1	1	0	<2
	0	0	1	1	1	1	<2
	0	1	0	1	1	1	<2
	1	1	1	1	1	2	>=2



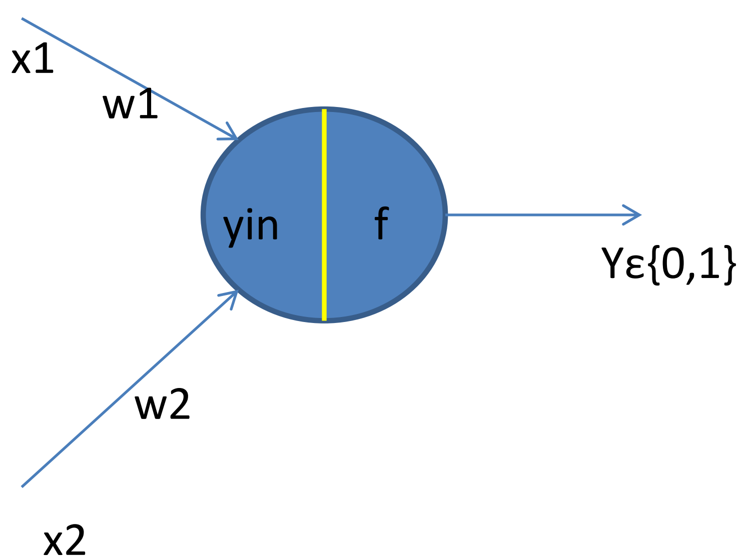


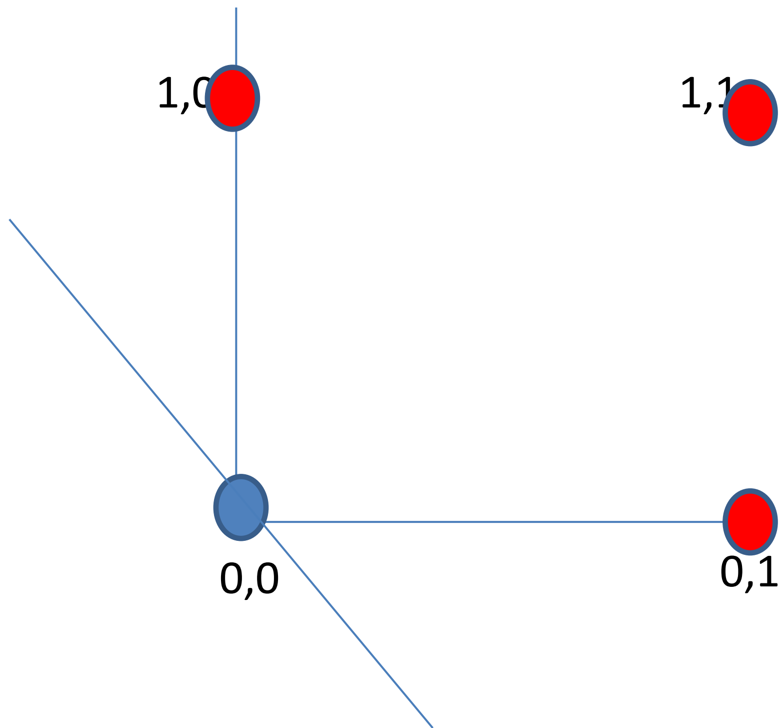
Gates	output	Inputs		Weights		yin	Threshold
OR	y	x1	x2	W1	w2		
	0	0	0	1	1	0	<1
	1	0	1	1	1	1	>=1
	1	1	0	1	1	1	>=1
	1	1	1	1	1	1	>=1



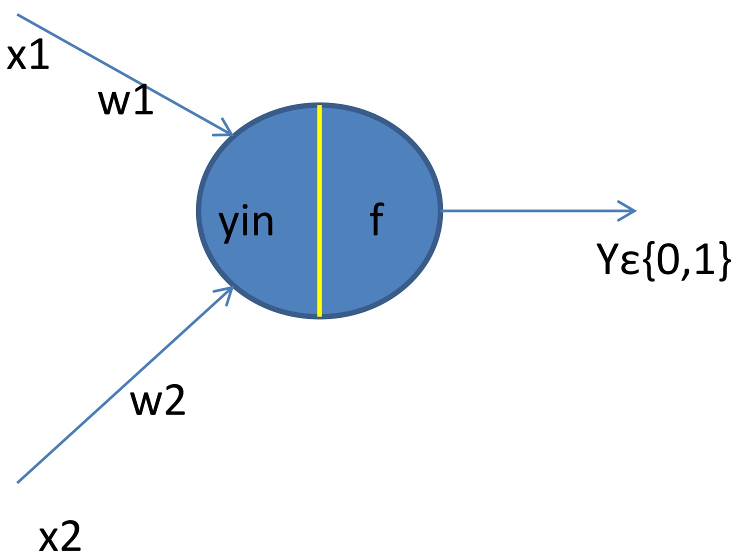


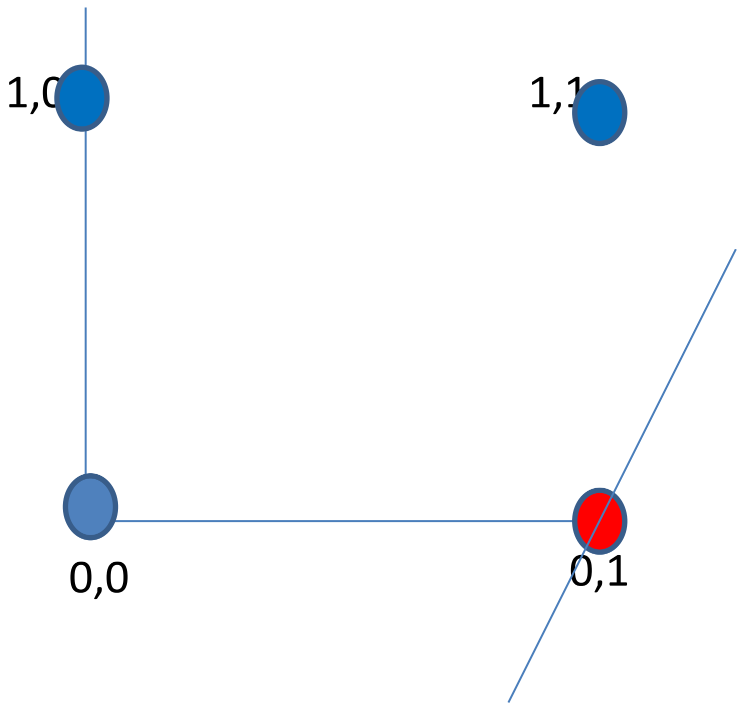
Gates	output	Inputs		Weights		yin	Threshold
NOR	y	x1	x2	W1	w2		
	1	0	0	-1	-1	0	>=0
	0	0	1	-1	-1	-1	<0
	0	1	0	-1	-1	-1	<0
	0	1	1	-1	-1	-2	<0





Gates	output	Inputs		Weights		yin	Threshold
ANDNOT	y	x1	x2	W1	w2		
	0	0	0	1	-1	0	<1
	0	0	1	1	-1	-1	<1
	1	1	0	1	-1	1	>=1
	0	1	1	1	-1	0	<1





EXOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Solve it using the Mcculloch pitts model

$$y = x_1 \overline{x_2} + \overline{x_1} x_2$$

$$y = z_1 + z_2$$

$$z_1 = x_1 \overline{x_2},\ z_2 = \overline{x_1} x_2,\ y = z_1 + z_2,$$

EXOR Contd...

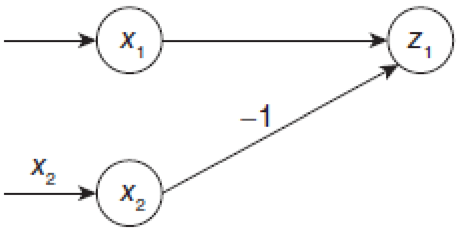
A single layer net is not sufficient to represent the function. An intermediate layer is necessary.

First function $(z_1 = x_1\overline{x}_2)$:

The truth table for function z_1 is shown as

x_1	x_2	z_1
0	0	0
0	1	0
1	0	1
1	1	0

1. Find out the case when both the weights are excitatory, both weight are inhibitory and one of then as inhibitory and choose the correct weights



EXOR Contd...

Second Function $\left(z_2 = \bar{x}_1 x_2\right)$:

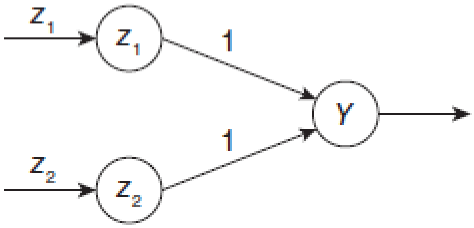
x_1	x_2	z_1
0	0	0
0	1	1
1	0	0
1	1	0

Third Function ($y = z_1$ OR z_2):

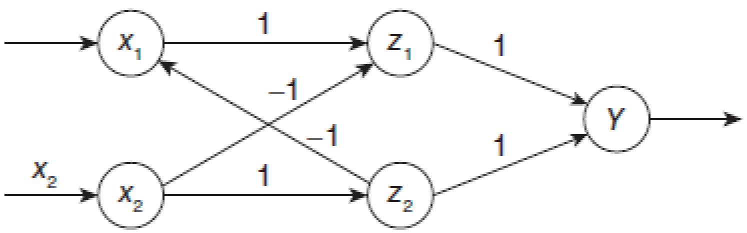
The truth table is given as

x_1	x_2	y	z_1	z_1
0	0	0	0	0
0	1	1	0	1
1	0	1	1	0
1	1	0	0	0

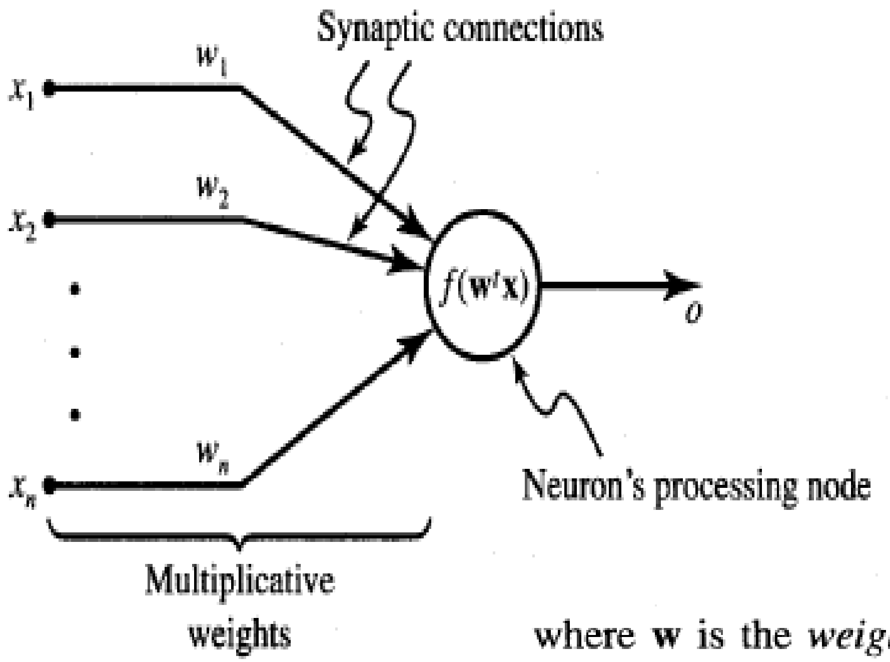
By setting threshold $\Theta \geq 1$, the network can be implemented.



The McCulloch–Pitts model for XOR function is given as follows:



General symbol of neuron consisting of processing node and synaptic connections



$$o = f(\mathbf{w}^t\mathbf{x}), \quad \text{or}$$

$$o = f\left(\sum_{i=1}^{n} w_i x_i\right)$$

where $\mathbf{w}$ is the *weight vector* defined as

$$\mathbf{w} \triangleq [w_1 \quad w_2 \quad \cdots \quad w_n]^t$$

and $\mathbf{x}$ is the input vector:

$$\mathbf{x} \triangleq [x_1 \quad x_2 \quad \cdots \quad x_n]^t$$

Neuron Modeling for ANN

$$o = f(\mathbf{w}^t\mathbf{x}), \text{ or}$$

$$o = f\left(\sum_{i=1}^{n} w_i x_i\right)$$

Is referred to activation function.
Domain is set of activation values *net.*

$$net \overset{\Delta}{=} \mathbf{w}^t\mathbf{x}$$

Scalar product of weight and input vector

Neuron as a processing node performs the operation of summation of its weighted input.

Activation function

- Bipolar binary and unipolar binary are called as **hard limiting** activation functions used in discrete neuron model
- Unipolar continuous and bipolar continuous are called **soft limiting** activation functions are called <u>sigmoidal</u> characteristics.

Activation functions

Bipolar continuous

$$f(net) \overset{\Delta}{=} \frac{2}{1 + \exp(-\lambda net)} - 1$$

$$\lambda > 0$$

$$f(net) \overset{\Delta}{=} \operatorname{sgn}(net) = \begin{cases} +1, & net > 0 \\ -1, & net < 0 \end{cases}$$

Bipolar binary functions

Activation functions

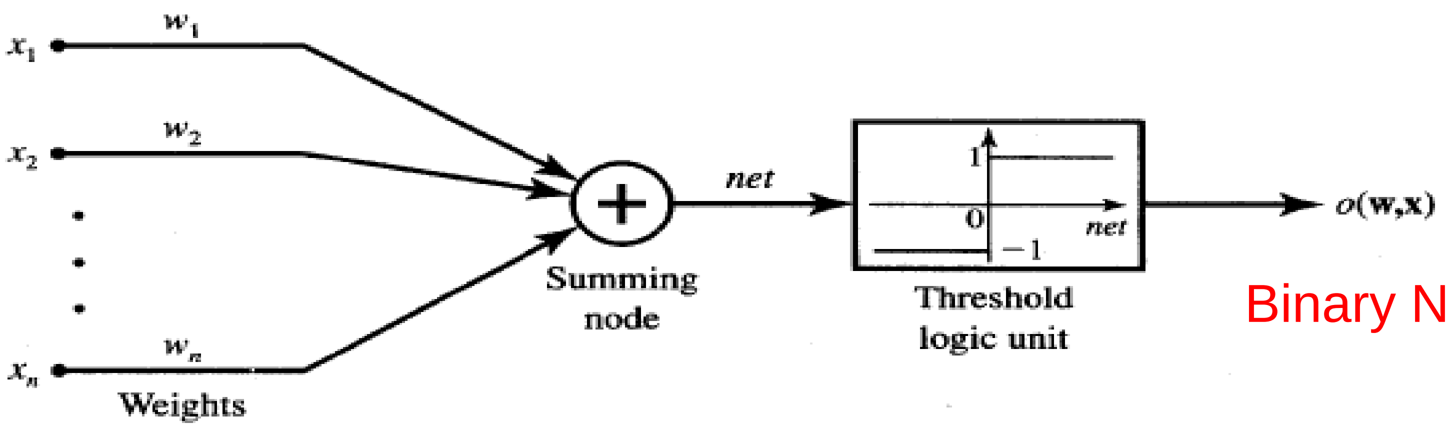
Unipolar continuous

$$f(net) \triangleq \frac{1}{1+\exp(-\lambda net)}$$

Unipolar Binary

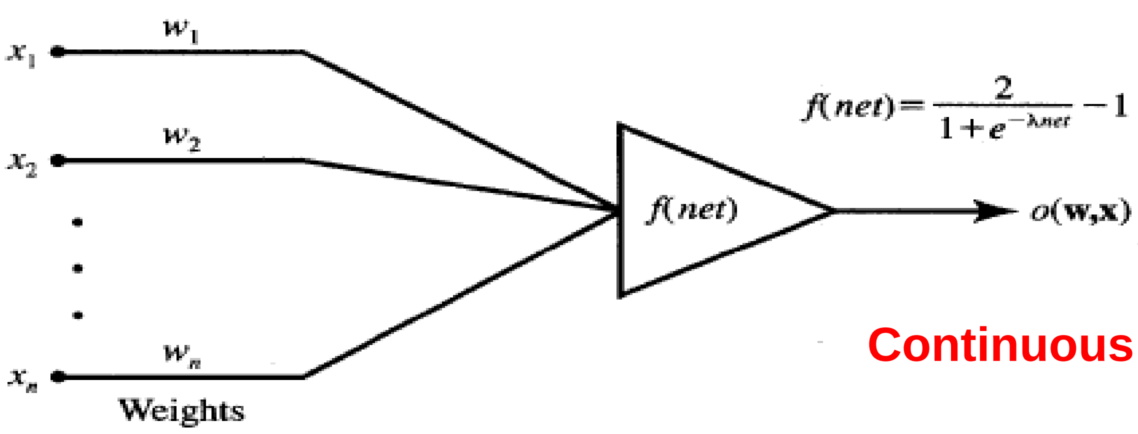
$$f(net) \triangleq \begin{cases} 1, & net > 0 \\ 0, & net < 0 \end{cases}$$

Common models of neurons



(a)

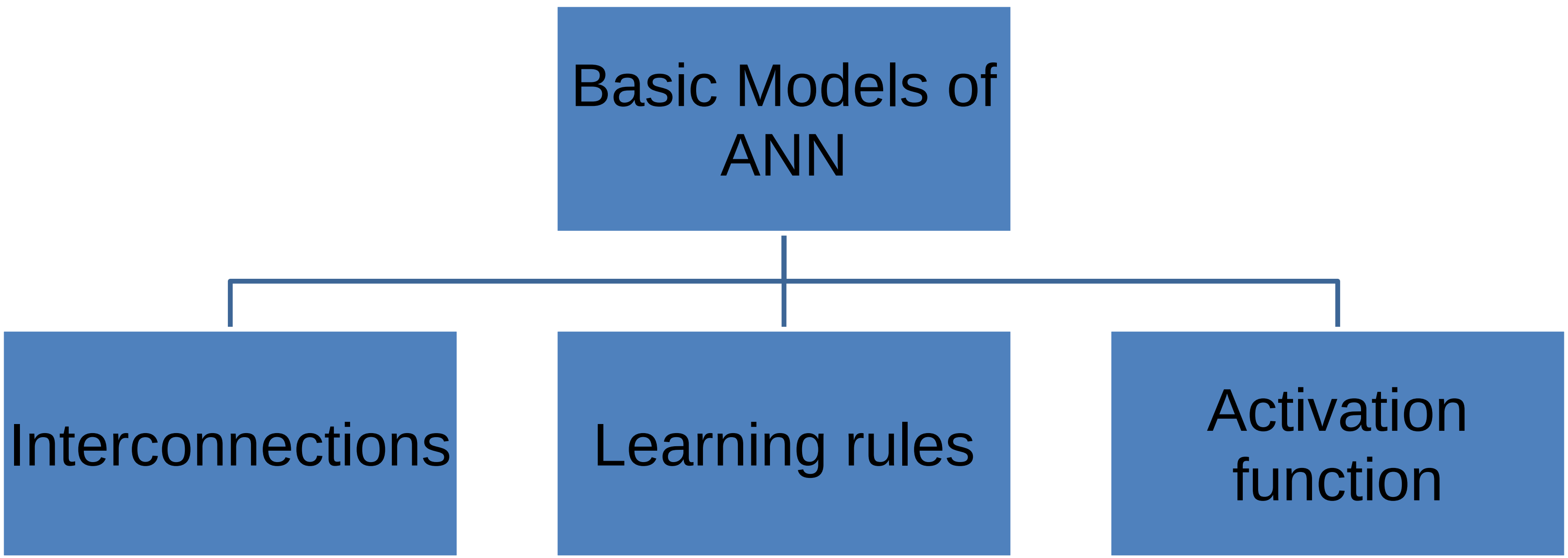
Binary Neurons



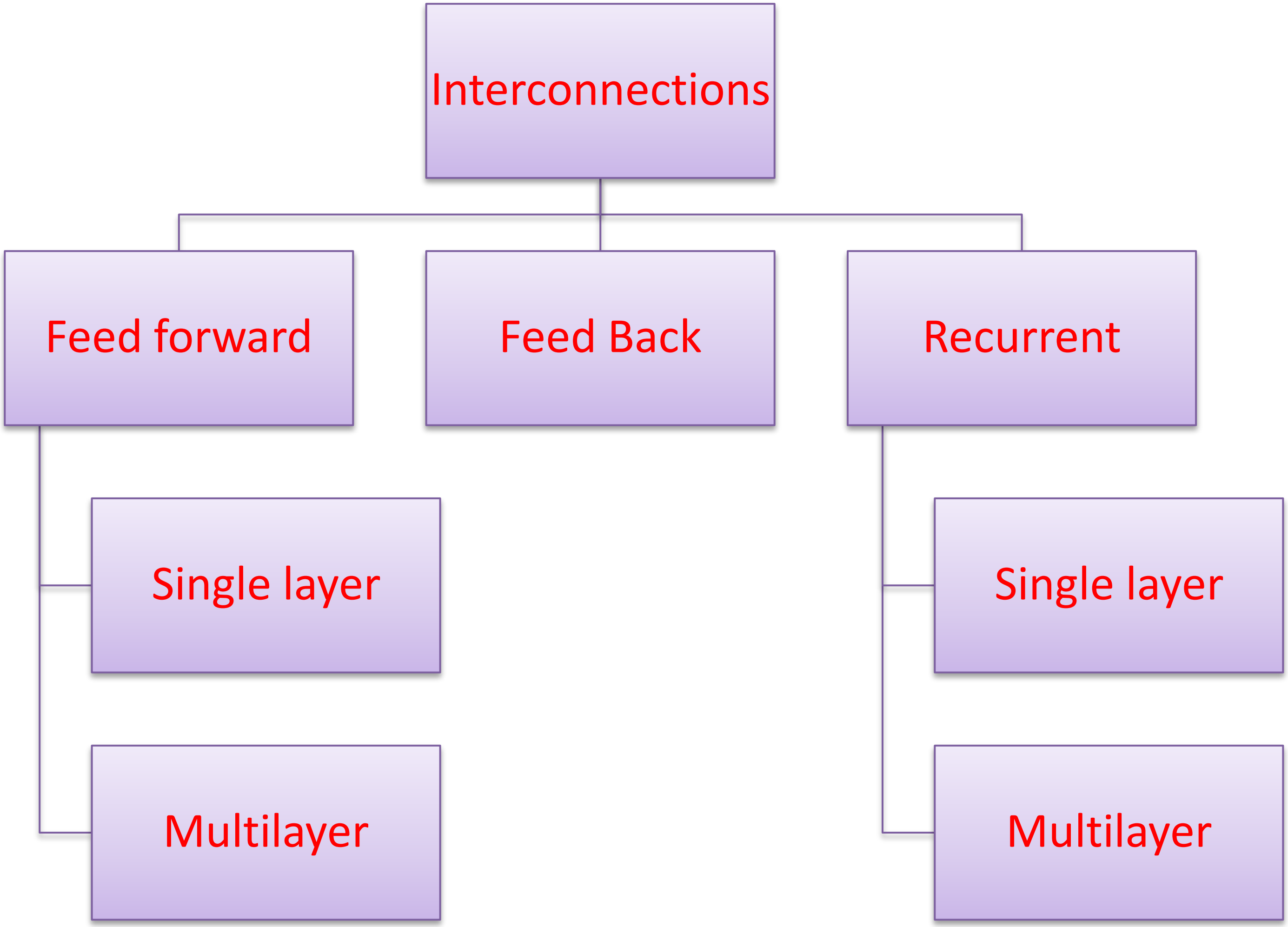
(b)

Continuous Neurons

Basic models of ANN

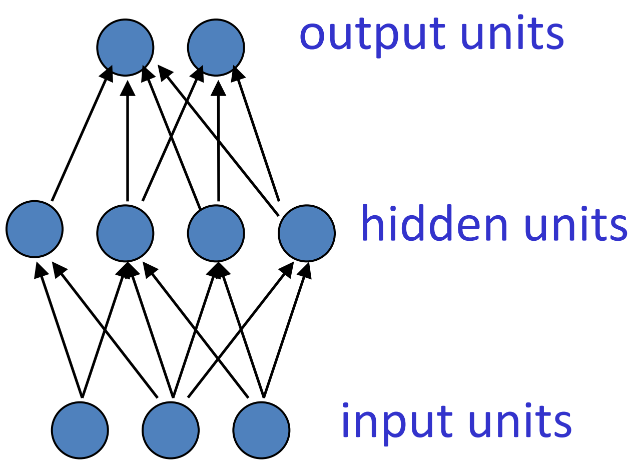


Classification based on interconnections



Feed-forward neural networks

- These are the commonest type of neural network in practical applications.
  - The first layer is the input and the last layer is the output.
  - If there is more than one hidden layer, we call them "deep" neural networks.
- They compute a series of transformations that change the similarities between cases.
  - The activities of the neurons in each layer are a non-linear function of the activities in the layer below.



Feedforward Network

- Its output and input vectors are respectively

$$\mathbf{o} = \begin{bmatrix} o_1 & o_2 & \cdots & o_m \end{bmatrix}^t$$
$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^t$$

- Weight w_{ij} connects the *i'th* neuron with *j'th* input. Activation rule of ith neuron is

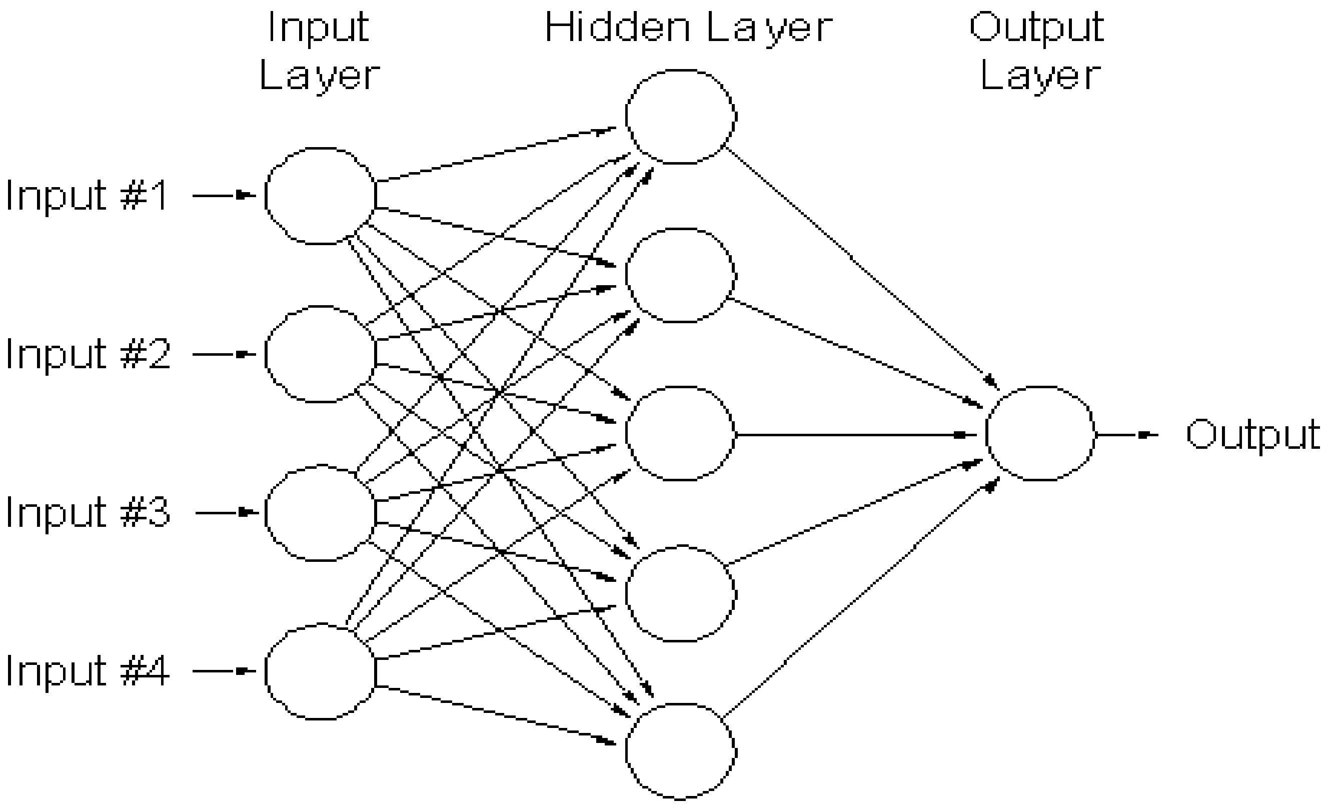
$$net_i = \sum_{j=1}^{n} w_{ij}x_j, \quad \text{for } i = 1, 2, \ldots, m$$

$$o_i = f(\mathbf{w}_i^t\mathbf{x}), \quad \text{for } i = 1, 2, \ldots, m$$

where

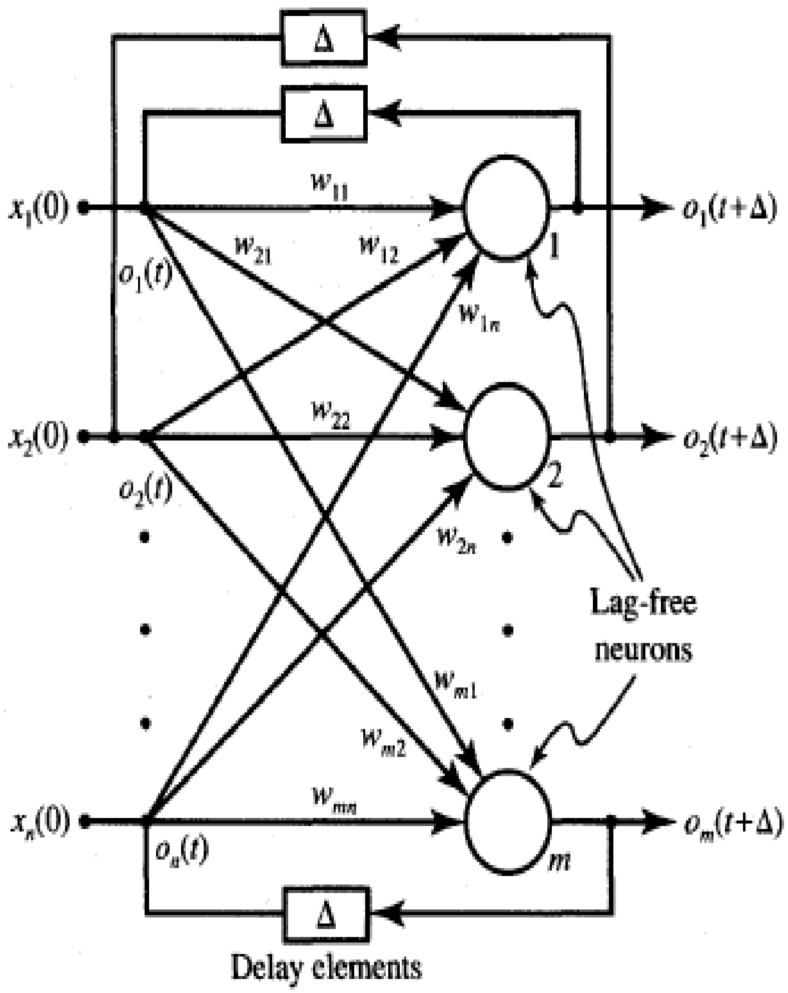
$$\mathbf{w}_i \triangleq \begin{bmatrix} w_{i1} & w_{i2} & \cdots & w_{in} \end{bmatrix}^t$$

Multilayer feed forward network

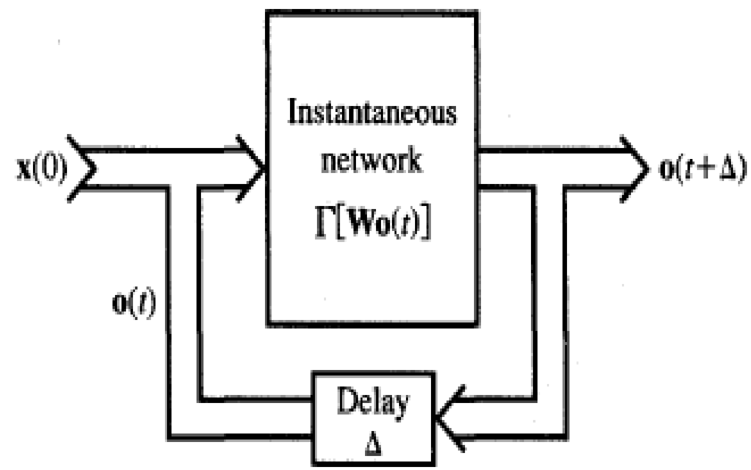


Can be used to solve complicated problems

Feedback network

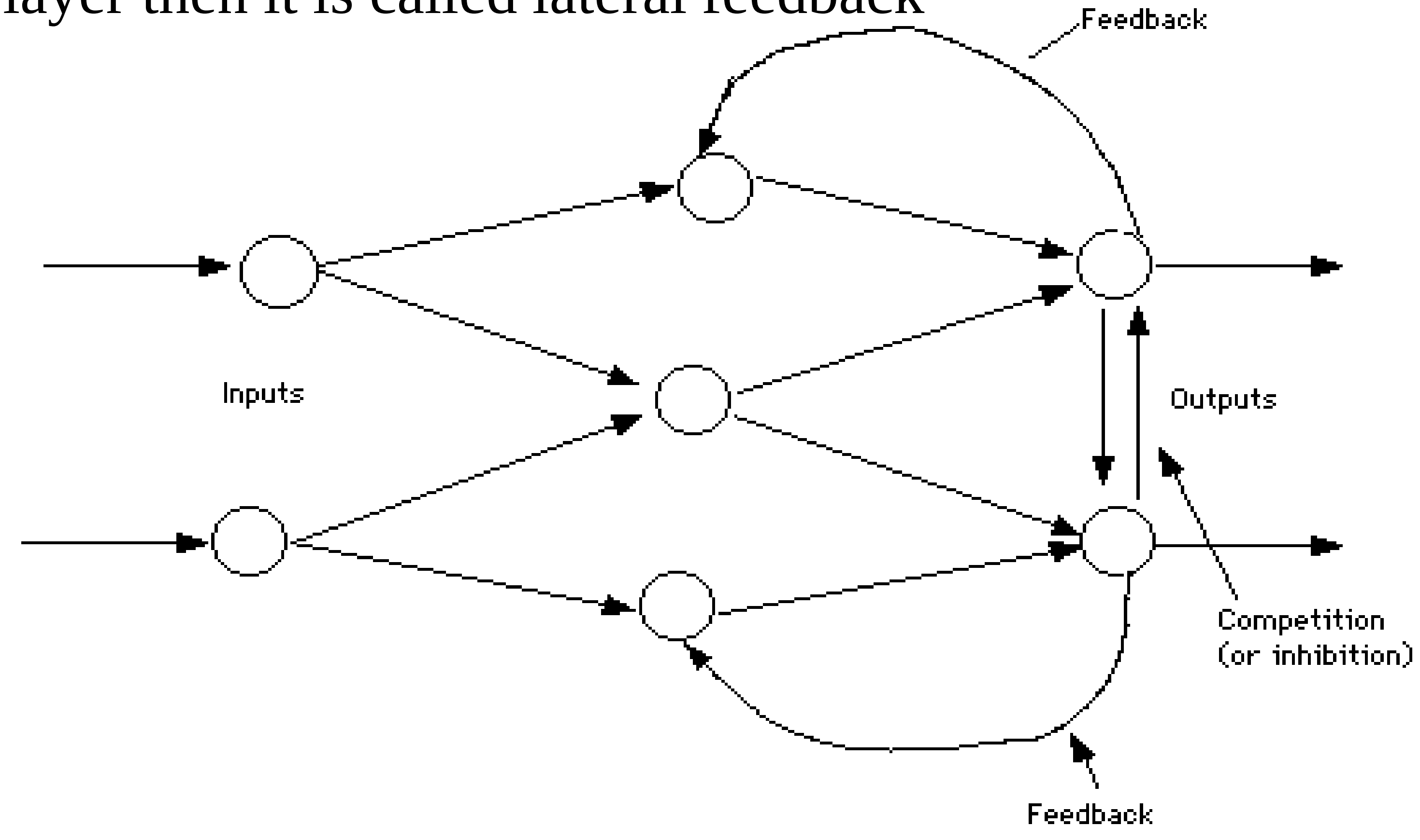


When outputs are directed back as inputs to same or preceding layer nodes it results in the formation of feedback networks



Lateral feedback

If the feedback of the output of the processing elements is directed back as input to the processing elements in the same layer then it is called lateral feedback

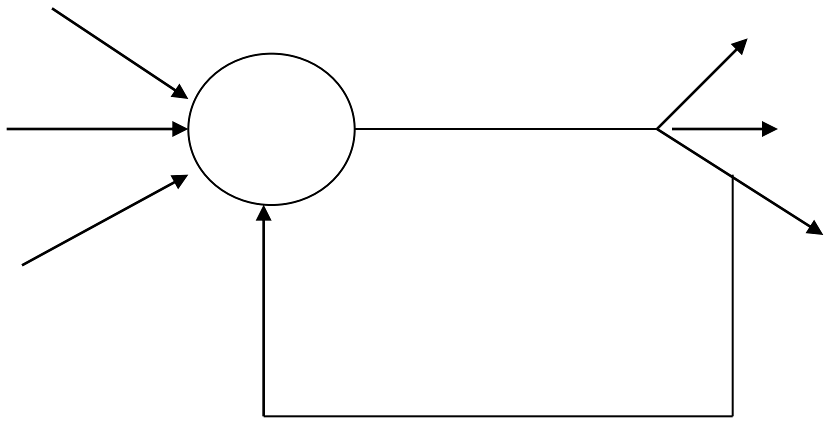


Recurrent n/ws

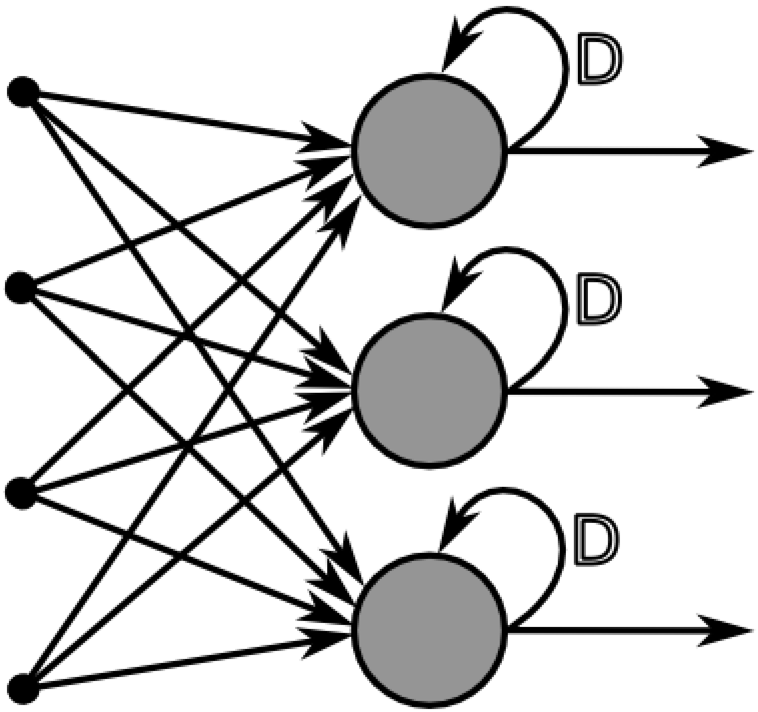
Feedback networks with closed loop are called **Recurrent Networks**. The response at the k+1'th instant depends on the entire history of the network starting at k=0.

- Single node with own feedback
- Competitive nets
- Single-layer recurrent networks
- Multilayer recurrent networks

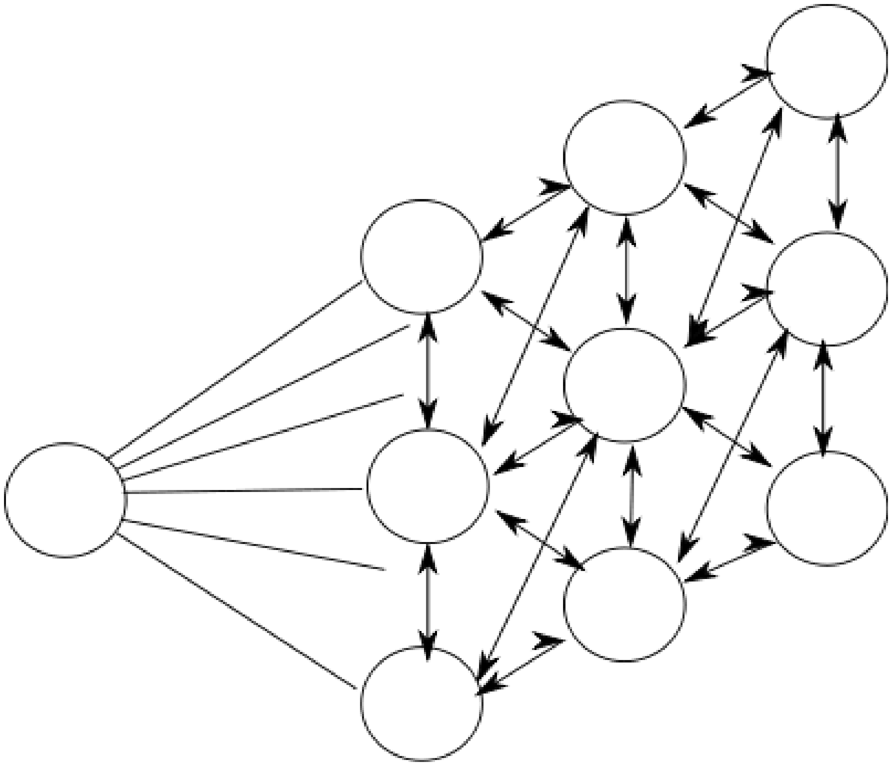
Single node with own feedback



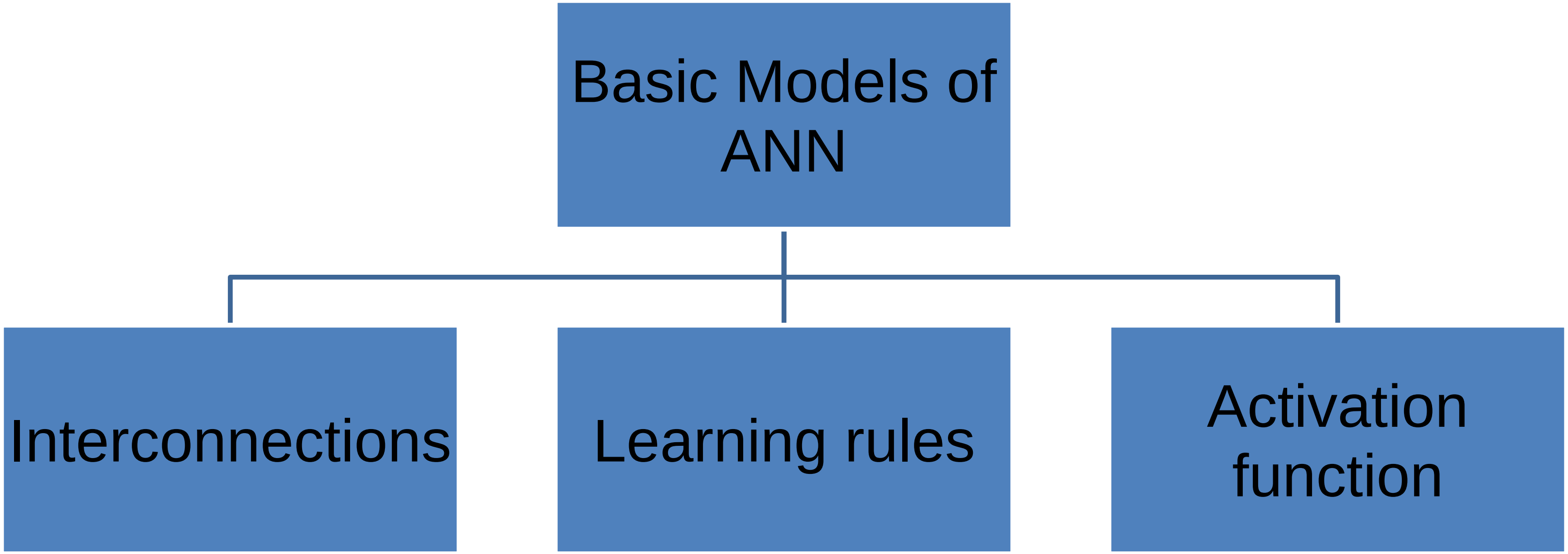
Single layer Recurrent Networks



Competitive networks



Basic models of ANN



Learning

- It’s a process by which a NN adapts itself to a stimulus by making proper parameter adjustments, resulting in the production of desired response
- Two kinds of learning
  - Parameter learning:- connection weights are updated
  - Structure Learning:- change in network structure

Training

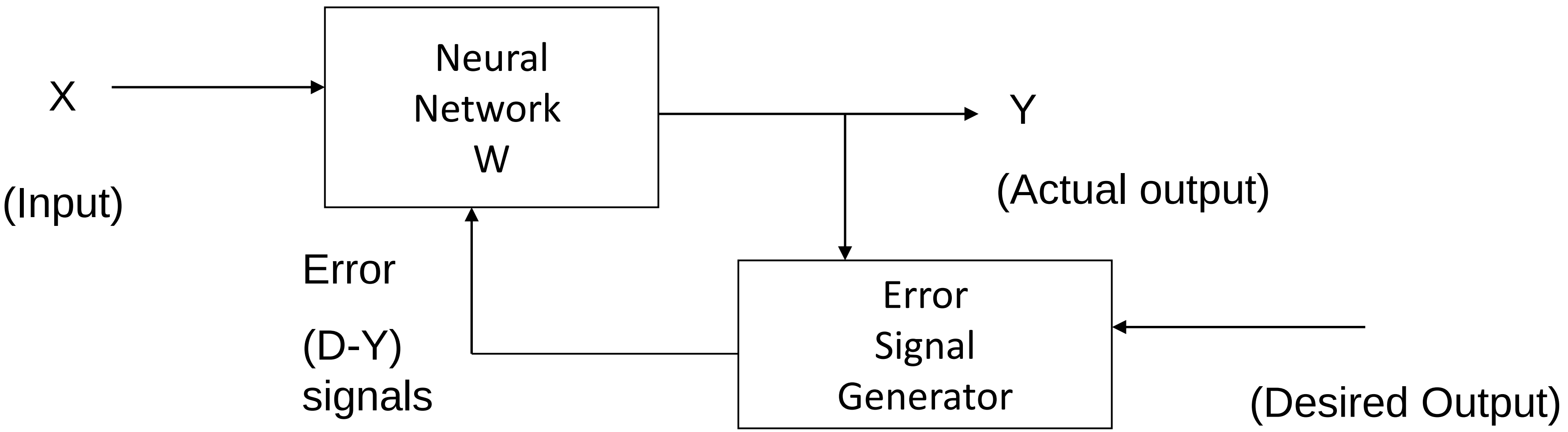
- The process of modifying the weights in the connections between network layers with the objective of achieving the expected output is called training a network.
- This is achieved through
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning

Classification of learning

- Supervised learning:-
  - Learn to predict an output when given an input vector.
- Unsupervised learning
  - Discover a good internal representation of the input.
- Reinforcement learning
  - Learn to select an action to maximize payoff.

Supervised Learning

- Child learns from a teacher
- Each input vector requires a corresponding target vector.
- Training pair=[input vector, target vector]



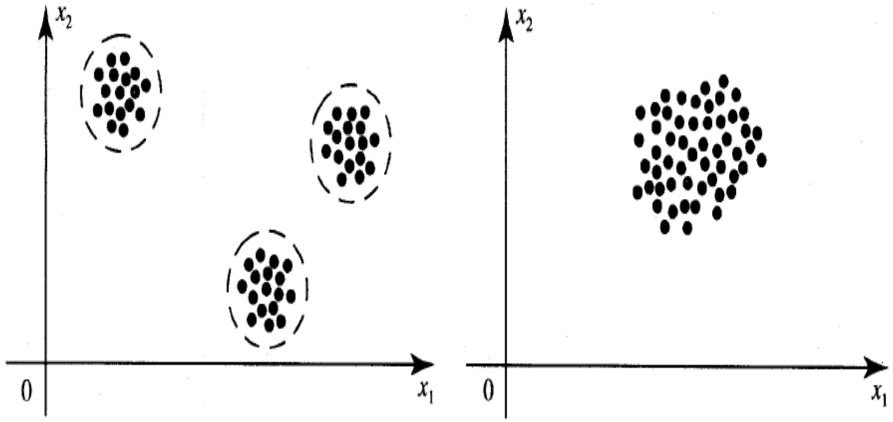
Two types of supervised learning

- Each training case consists of an input vector x and a target output t.

- **Regression**: The target output is a real number or a whole vector of real numbers.
  - The price of a stock in 6 months time.
  - The temperature at noon tomorrow.

- **Classification**: The target output is a class label.
  - The simplest case is a choice between 1 and 0.
  - We can also have multiple alternative labels.

Unsupervised Learning

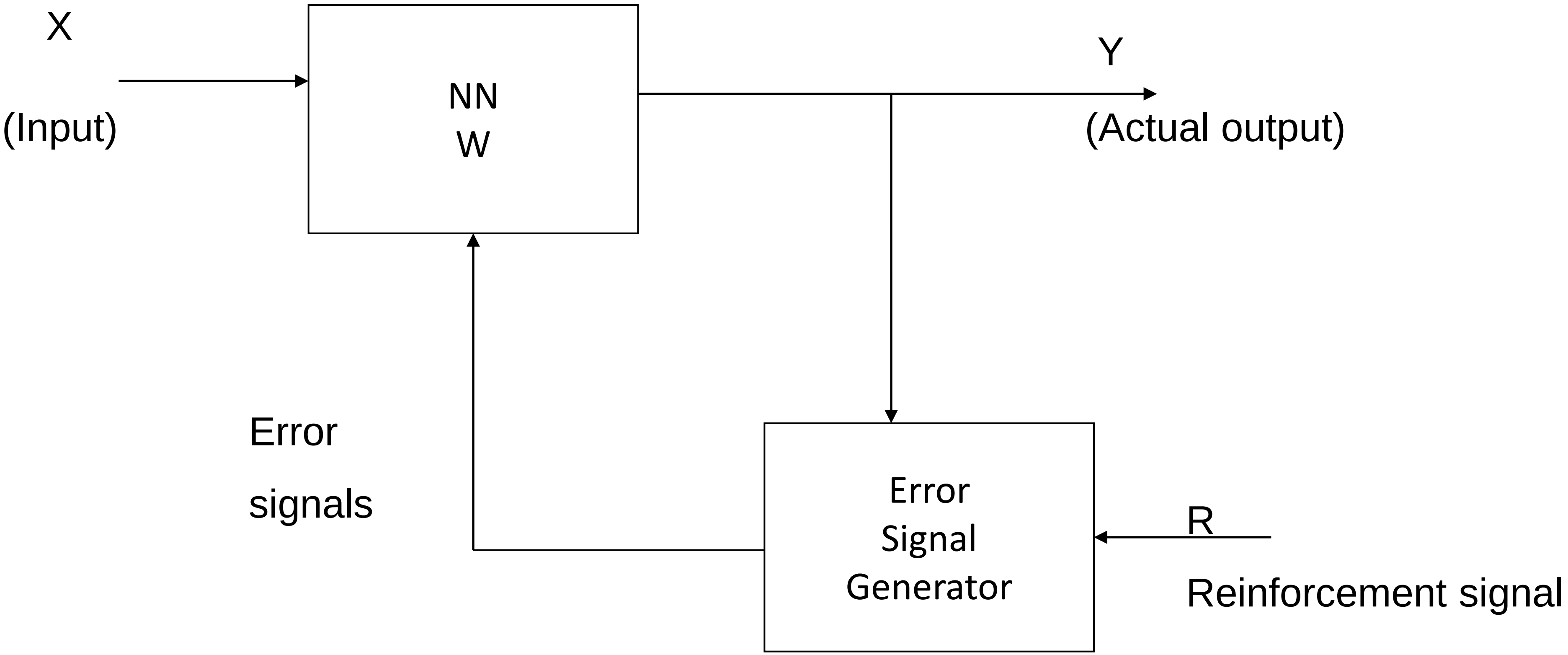


- All similar input patterns are grouped together as clusters.
- If a matching input pattern is not found a new cluster is formed
- Unsupervised classification of patterns/objects without providing information about the actual classes

Self-organizing

- In unsupervised learning there is no feedback
- Network must discover patterns, regularities, features for the input data over the output
- While doing so the network might change in parameters
- This process is called **self-organizing.**

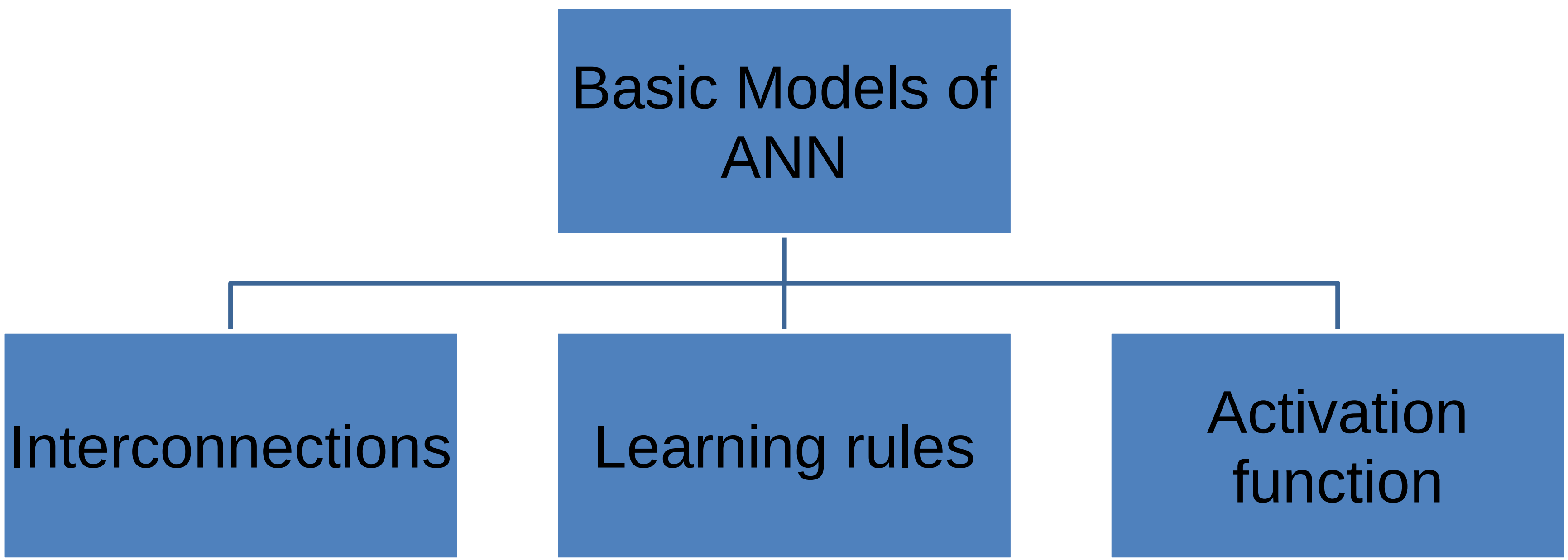
Reinforcement Learning



When Reinforcement learning is used?

- If less information is available about the target output values (critic information)
- Learning based on this critic information is called reinforcement learning and the feedback sent is called reinforcement signal

Basic models of ANN



Activation Function

1. Identity Function

   f(x)=x for all x
2. Binary Step function

$$f(x) = \begin{cases} 1 \, if x \geq \theta \\ 0 \, if x < \theta \end{cases}$$

3. Bipolar Step function

$$f(x) = \begin{cases} 1 \, if x \geq \theta \\ -1 \, if x < \theta \end{cases}$$

4. Sigmoidal Functions:- Continuous functions
5. Ramp functions:-

$$f(x) = \begin{cases} 1 \, if x > 1 \\ x \, if \, 0 \leq x \leq 1 \\ 0 \, if x < 0 \end{cases}$$

Activation functions

Types of Activation Function:

- Threshold Function
- Piecewise-Linear Function
- Sigmoid Function (signum fuction or hyperbolic tangent function)

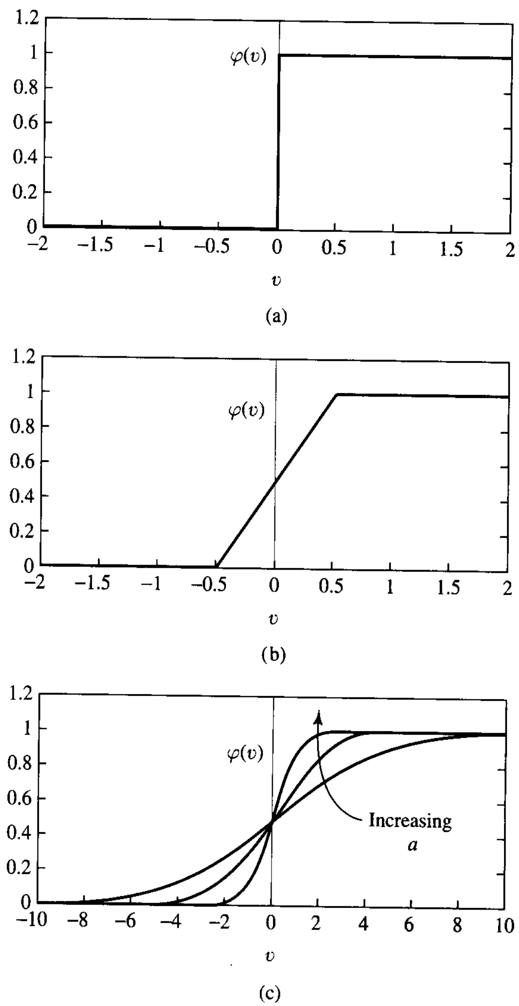


FIGURE 1.8 (a) Threshold function. (b) Piecewise-linear function. (c) Sigmoid function for varying slope parameter *a*.

Important terminologies of ANNs

- Weights
- Bias
- Threshold
- Learning rate

Weights

- Each neuron is connected to every other neuron by means of directed links
- Links are associated with weights
- Weights contain information about the input signal and is represented as a matrix
- Weight matrix also called <u>connection matrix</u>

Weight Matrix

$$W = \begin{bmatrix} w_1^T \\ w_2^T \\ w_3^T \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ w_n^T \end{bmatrix} = \begin{bmatrix} W_{11} W_{12} W_{13} \cdots W_{1m} \\ W_{21} W_{22} W_{23} \cdots W_{2m} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ W_{n1} W_{n2} W_{n3} \cdots W_{nm} \end{bmatrix}$$

Activation Functions

- Used to calculate the output response of a neuron.
- Sum of the weighted input signal is applied with an activation to obtain the response.
- Activation functions can be linear or non linear

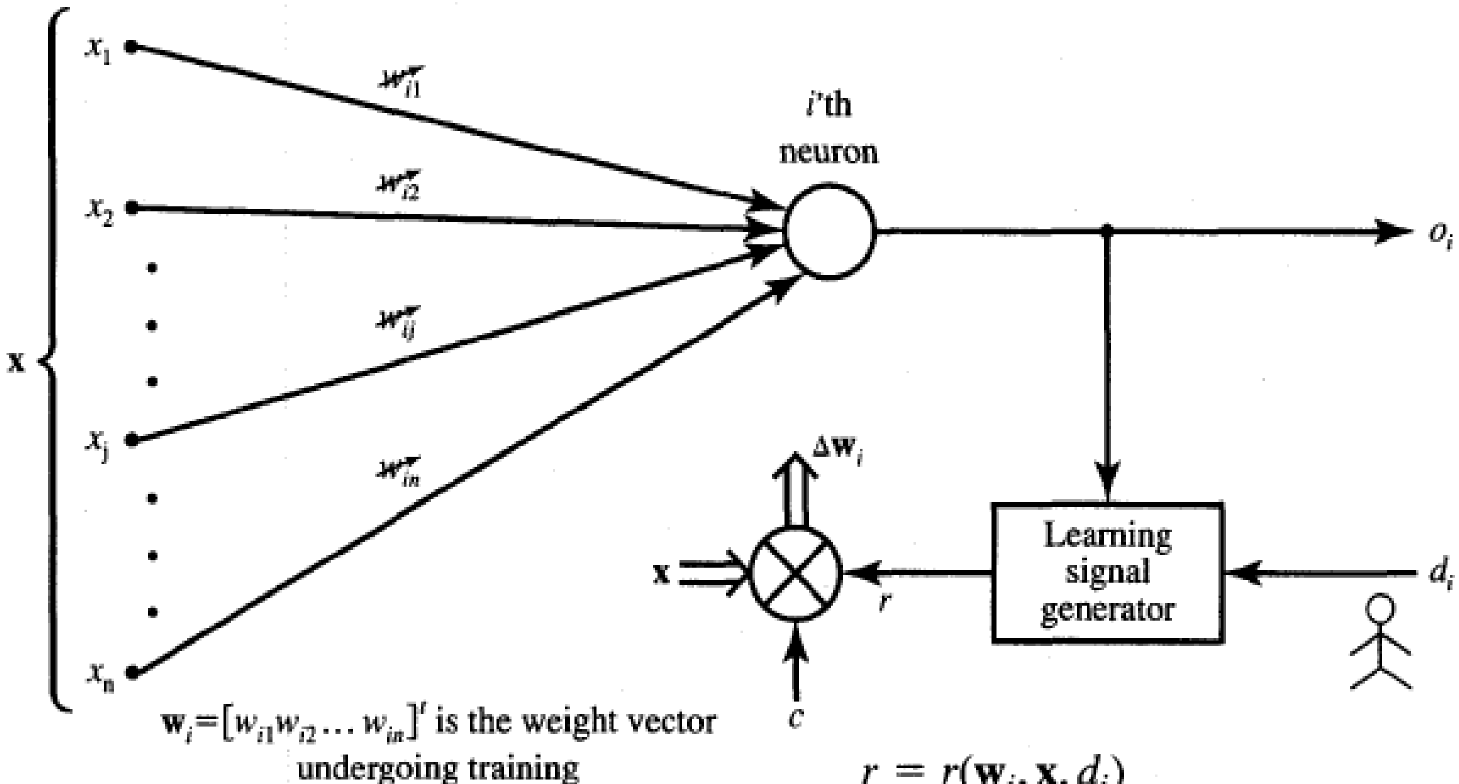
Bias

- Bias is like another weight. Its included by adding a component $x_0=1$ to the input vector X.
- $X=(1,X_1,X_2\ldots X_i,\ldots X_n)$
- Bias is of two types
  - Positive bias: increase the net input
  - Negative bias: decrease the net input

Neural Network Learning Rules

- The weight vector increases in proportion to the product of input x and learning signal r.

Neural Network Learning Rules



$$r = r(\mathbf{w}_i, \mathbf{x}, d_i)$$

$$\Delta \mathbf{w}_i(t) = cr\left[\mathbf{w}_i(t), \mathbf{x}(t), d_i(t)\right] \mathbf{x}(t)$$

***c* – learning constant**

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + cr\left[\mathbf{w}_i(t), \mathbf{x}(t), d_i(t)\right] \mathbf{x}(t)$$

Hebbian Learning Rule

FEED FORWARD UNSUPERVISED LEARNING

- The learning signal is equal to the <u>neuron's output</u>

$$r \triangleq f(\mathbf{w}_i^t \mathbf{x})$$

$$\Delta \mathbf{w}_i = c f(\mathbf{w}_i^t \mathbf{x}) \mathbf{x}$$

The single weight w_{ij} is adapted using the following increment:

$$\Delta w_{ij} = c f(\mathbf{w}_i^t \mathbf{x}) x_j$$

This can be written briefly as

$$\Delta w_{ij} = c o_i x_j, \quad \text{for } j = 1, 2, \ldots, n$$

Features of Hebbian Learning

- Feed forward unsupervised learning
- "When an axon of a cell A is near enough to excite a cell B and repeatedly and persistently takes place in firing it, some growth process or change takes place in one or both cells increasing the efficiency"
- If $o_i x_j$ is positive, results in increase in weight else vice versa

$$\mathbf{w}^1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$$

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$$

$$c = 1$$

$$f(net) = \operatorname{sgn}(net).$$

Solution of Hebbian Learning

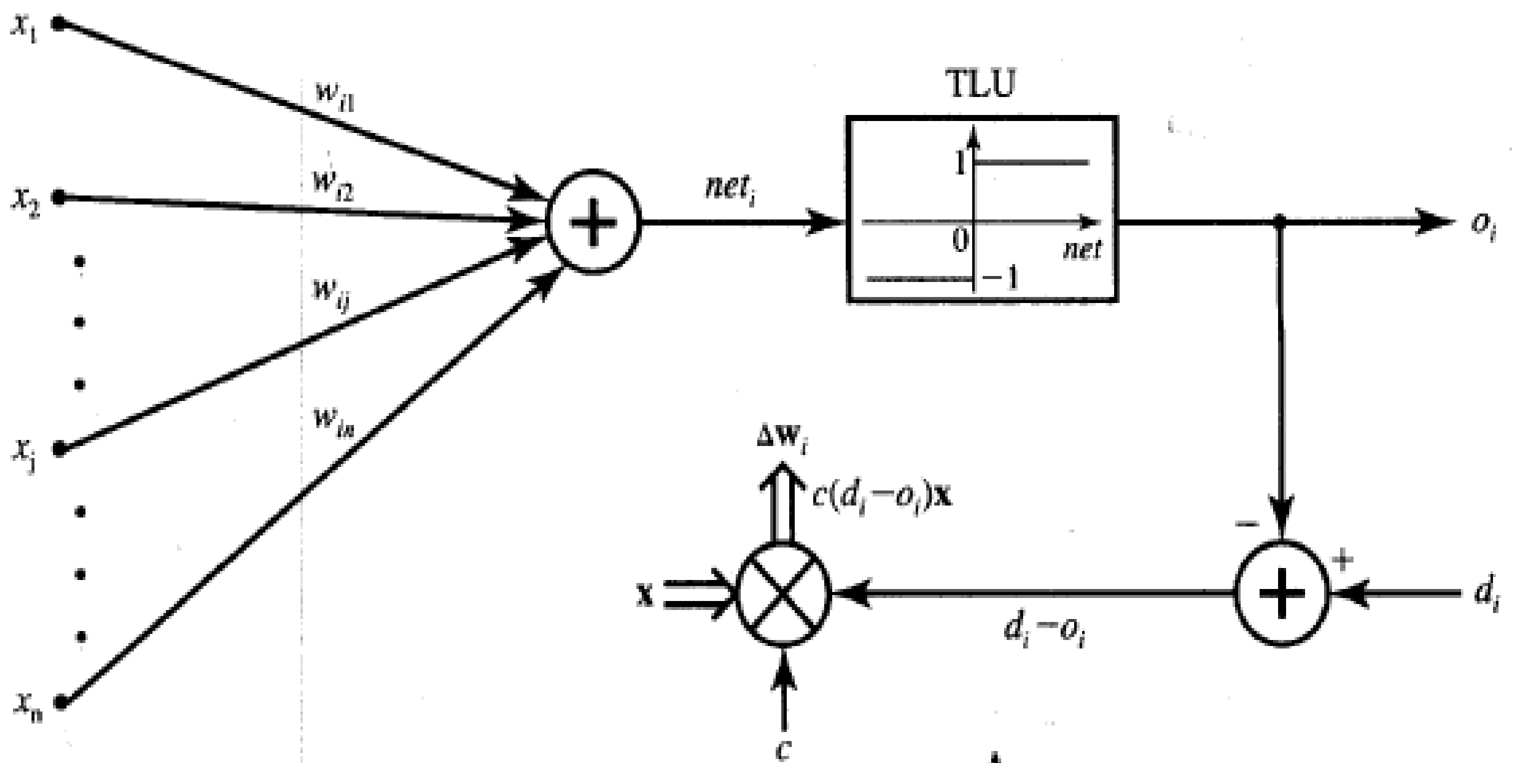
- Find the net input for the input x1
- Find f(net)
- Compute w2=w1+f(net)x1
- With w2 and x2 find net2
- Continue the above procedure till we get the final weights

$$\begin{bmatrix} 1 \\ -3.5 \\ 4.5 \\ 0.5 \end{bmatrix}$$

Perceptron Learning Rule

- Learning signal is the difference between the desired and actual neuron's response.
- Learning is **supervised.**
- Applicable only for **binary neuron** response.

Perceptron Learning Rule



$$r \triangleq d_i - o_i$$

$$o_i = \operatorname{sgn}(\mathbf{w}_i^t \mathbf{x})$$

$$\Delta \mathbf{w}_i = c \left[d_i - \operatorname{sgn}(\mathbf{w}_i^t \mathbf{x}) \right] \mathbf{x}$$

Example

This example illustrates the perceptron learning rule of the network shown in Figure 2.23. The set of input training vectors is as follows:

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix} \quad \mathbf{w}^1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$$

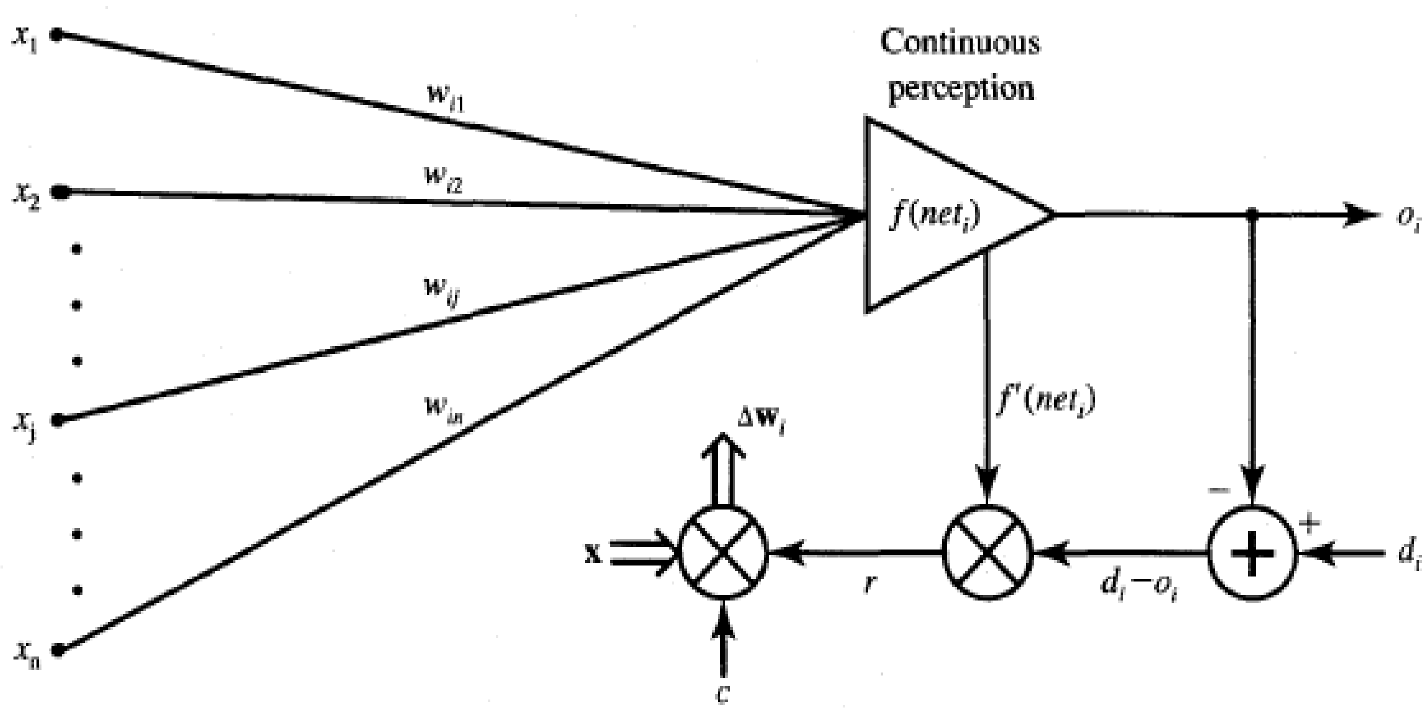
and the initial weight vector $\mathbf{w}^1$ is assumed identical as in Example 2.4. The learning constant is assumed to be $c = 0.1$. The teacher's desired responses for $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ are $d_1 = -1$, $d_2 = -1$, and $d_3 = 1$, respectively. The learning according to the perceptron learning rule progresses as follows.

$$\begin{bmatrix} 0.6 \\ -0.4 \\ 0.1 \\ 0.5 \end{bmatrix}$$

Delta Learning Rule

- Only valid for **continuous** activation function
- Used in supervised training mode
- Learning signal for this rule is called delta
- The aim of the delta rule is to minimize the error over all training patterns

$$r \triangleq [d_i - f(\mathbf{w}_i^t \mathbf{x})] f'(\mathbf{w}_i^t \mathbf{x})$$



Widrow-Hoff learning Rule

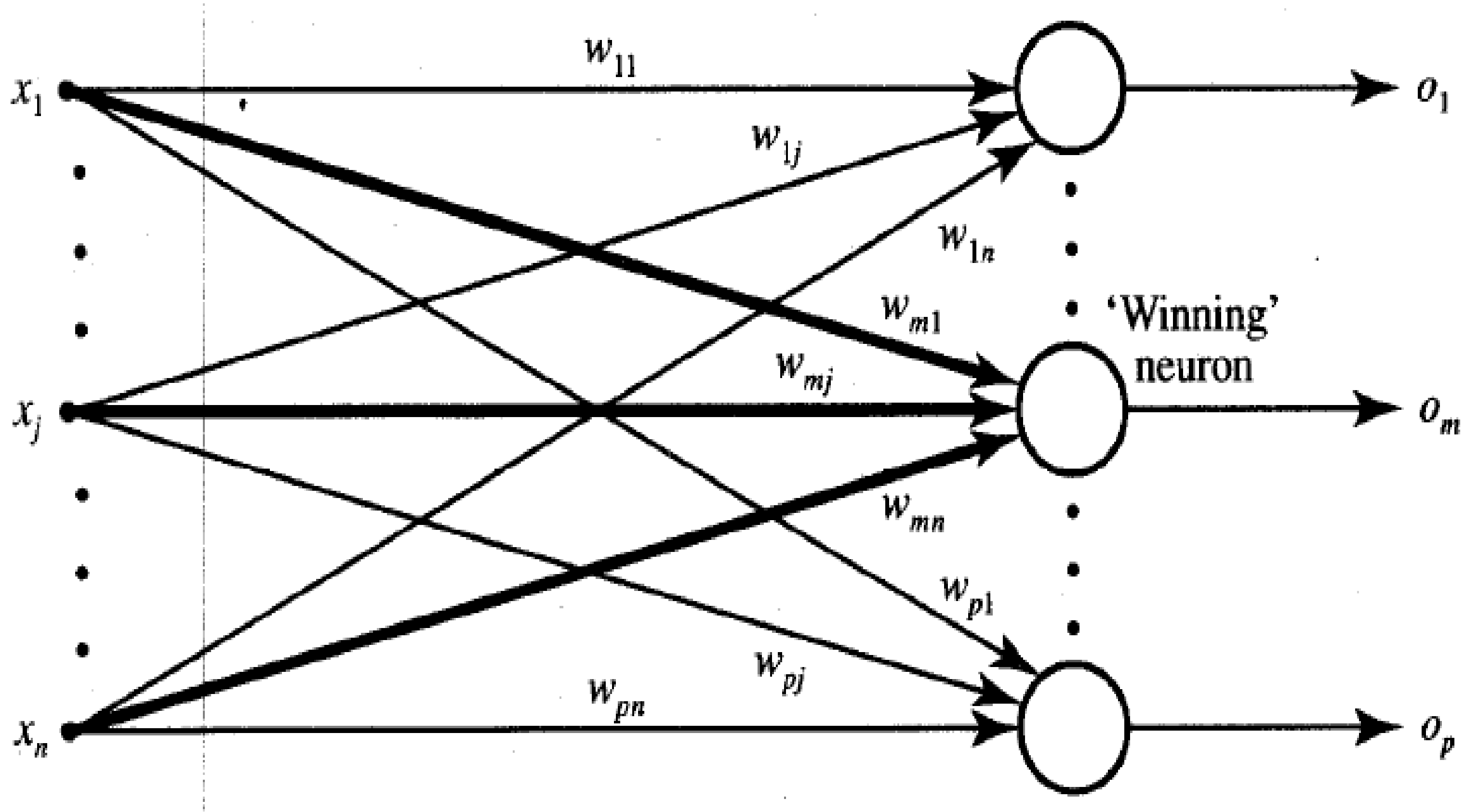
- Also called as least mean square learning rule
- Introduced by Widrow(1962), used in supervised learning
- Independent of the activation function
- Special case of delta learning rule wherein activation function is an identity function i.e. *f(net)=net*
- Minimizes the squared error between the desired output value d_i and net_i

$$r \triangleq d_i - \mathbf{w}_i^t \mathbf{x}$$

The weight vector increment under this learning rule is

$$\Delta \mathbf{w}_i = c(d_i - \mathbf{w}_i^t \mathbf{x})\mathbf{x}$$

Winner-Take-All learning rule



(adjusted weights are highlighted)

Winner-Take-All Learning rule Contd…

- Can be explained for a layer of neurons
- Example of competitive learning and used for unsupervised network training
- Learning is based on the premise that one of the neurons in the layer has a maximum response due to the input x
- This neuron is declared the winner with a weight

$$\mathbf{w}_m = \begin{bmatrix} w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix}^t$$

Its increment is computed as follows

$$\Delta \mathbf{w}_m = \alpha(\mathbf{x} - \mathbf{w}_m)$$

Contd..

The winner selection is based on the following criterion of maximum activation among all p neurons participating in a competition:

$$\mathbf{w}_m^t \mathbf{x} = \max_{i=1,2,\ldots,p} (\mathbf{w}_i^t \mathbf{x})$$

Summary of learning rules

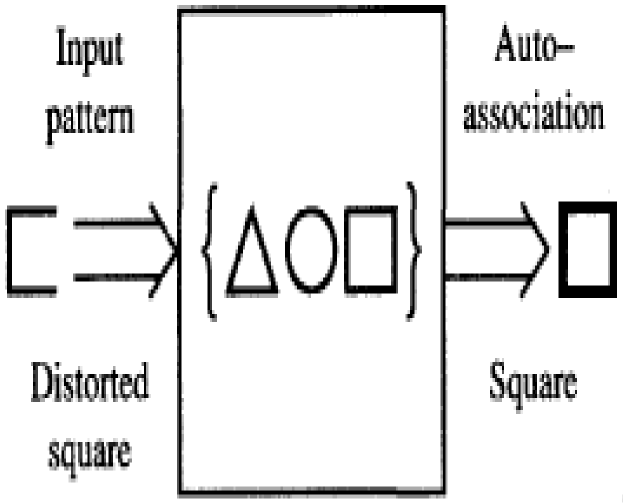
Summary of learning rules and their properties.

Learning rule	Single weight adjustment Δw_{ij}	Initial weights	Learning	Neuron characteristics	Neuron / Layer
Hebbian	$co_i x_j$ $j = 1, 2, \ldots, n$	0	U	Any	Neuron
Perceptron	$c\left[d_i - \operatorname{sgn}(\mathbf{w}_i^t \mathbf{x})\right] x_j$ $j = 1, 2, \ldots, n$	Any	S	Binary bipolar, or Binary unipolar*	Neuron
Delta	$c(d_i - o_i) f'(net_i) x_j$ $j = 1, 2, \ldots, n$	Any	S	Continuous	Neuron
Widrow-Hoff	$c(d_i - \mathbf{w}_i^t \mathbf{x}) x_j$ $j = 1, 2, \ldots, n$	Any	S	Any	Neuron
Winner-take-all	$\Delta w_{mj} = \alpha(x_j - w_{mj})$ m-winning neuron number $j = 1, 2, \ldots, n$	Random Normalized	U	Continuous	Layer of p neurons

Neural Processing

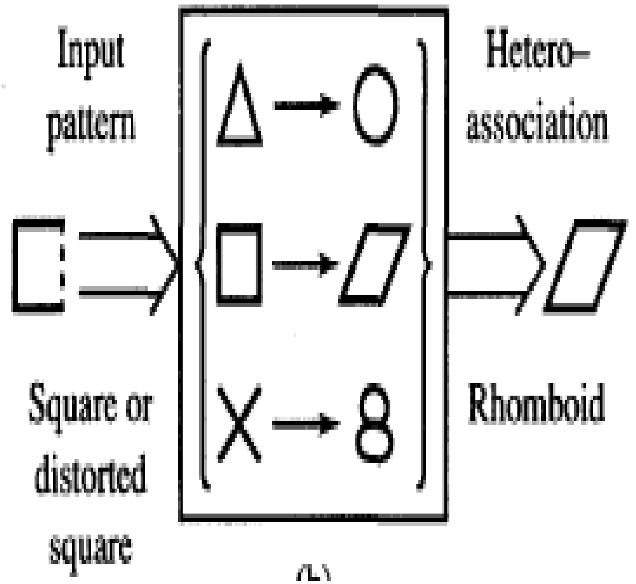
- **Recall:-** Processing phase for a NN and its objective is to retrieve the information. The process of computing **o** for a given **x.**
- **Basic forms of neural information processing**
  - Auto association
  - Hetero association
  - Classification

Neural Processing- Autoassociation



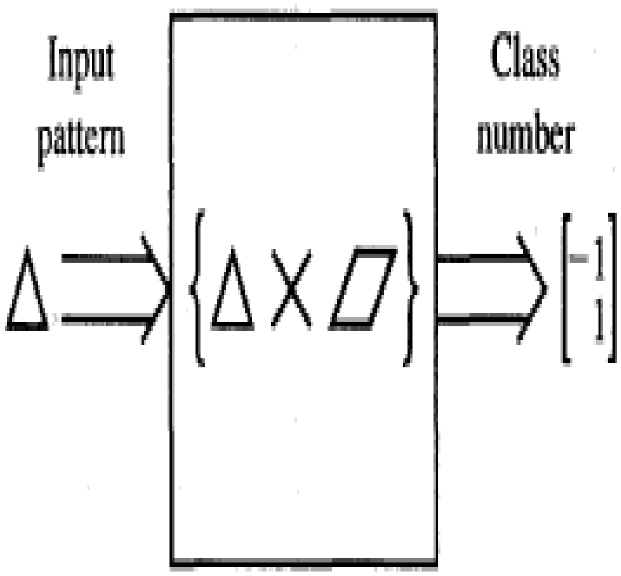
- Set of patterns can be stored in the network
- If a pattern similar to a member of the stored set is presented, an association with the input of closest stored pattern is made

Neural Processing- Heteroassociation



- Associations between pairs of patterns are stored
- Distorted input pattern may cause correct heteroassociation at the output

Neural Processing-Classification



- Set of input patterns is divided into a number of classes or categories
- In response to an input pattern from the set, the classifier is supposed to recall the information regarding class membership of the input pattern.

Quick review and list of important questions

- Evolution of Neural Networks
- Basics of ANN
- Biological neuron
- Basics of ANN
- Types of activation function
- Problems on activation function
- McCulloch Pitts model and solving numerical based on it

- Types of Learning rules
  - Hebbian
  - Perceptron
  - Delta
  - Widrow hoff
  - Winner take all
- Types of Neural processing