Artificial Neural Networks

Artificial Neural Network

ANN is a computing system made up of a number of simple, *highly interconnected processing elements*, which process information by their dynamic state response to external *inputs*.

Dr. Robert Hecht-Nielson as quoted in "Neural Network Primer: Part I" by Maureen Caudill, Ai Expert, Feb. 1989

- ANNs are modeled on the ***parallel architecture of animal / human brains***.
- The network is based on a simple form of ***inputs*** and ***outputs***.

Why Artificial Neural Networks?

ANN modeled
(inspired by biological neurons)

- ***Create Intelligence to solve complex problems***

Massive parallelism, Low energy consumption

Learning ability, Fault tolerance

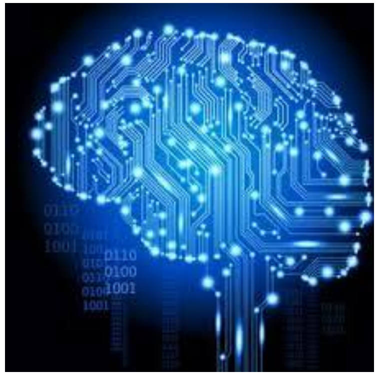
Characteristics of a human brain

Adaptivity (self-organization)

Generalization ability, Contextual information processing

not present in the modern computers

Modern Machine



Three periods of development for ANN

1940’s

McCulloch and Pitts – Simple Neural Network Model

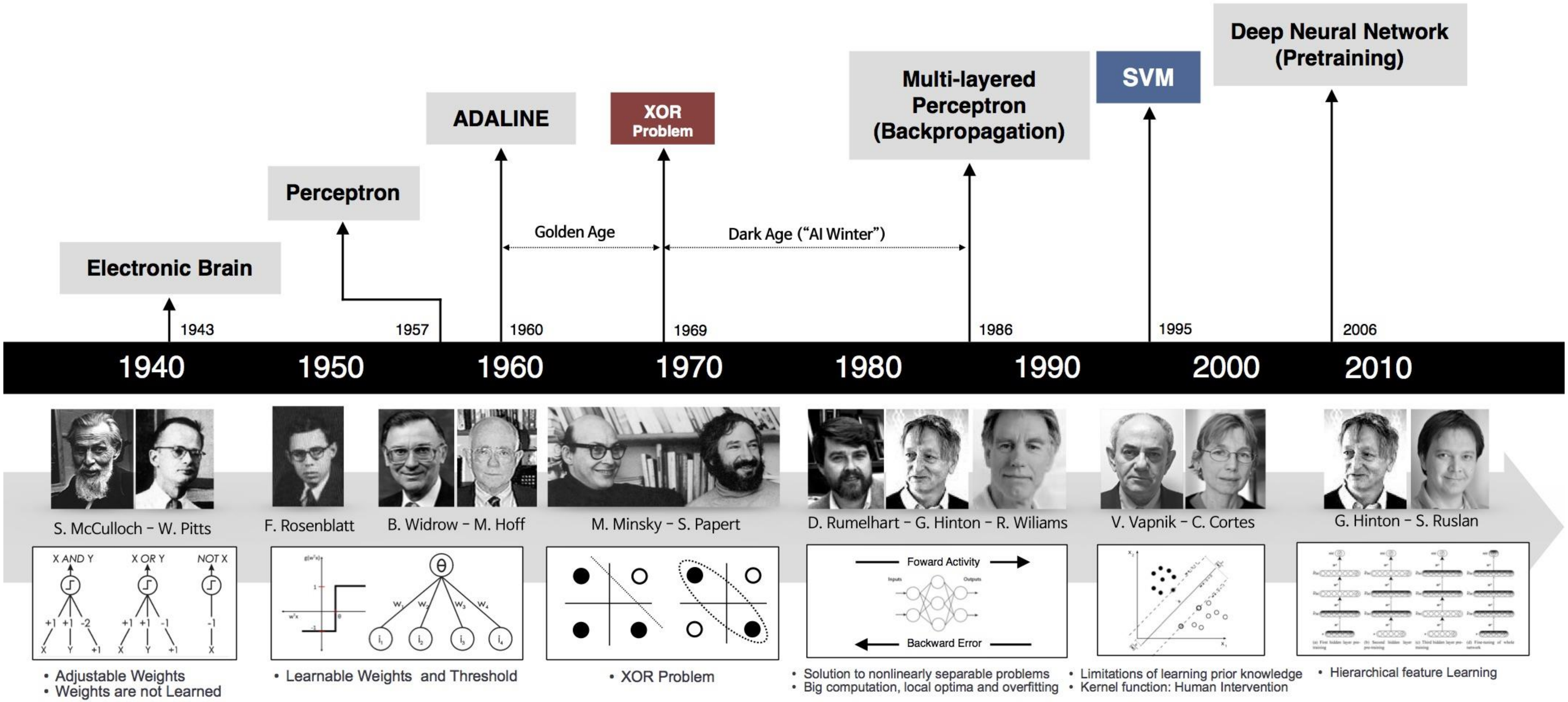
In **1960’s**

Rosenblatt - Perceptron Model

In 1980’s

Hopfield and Rumelhart - Hopfield's and Back-propagation Models

History of Artificial Neural Networks



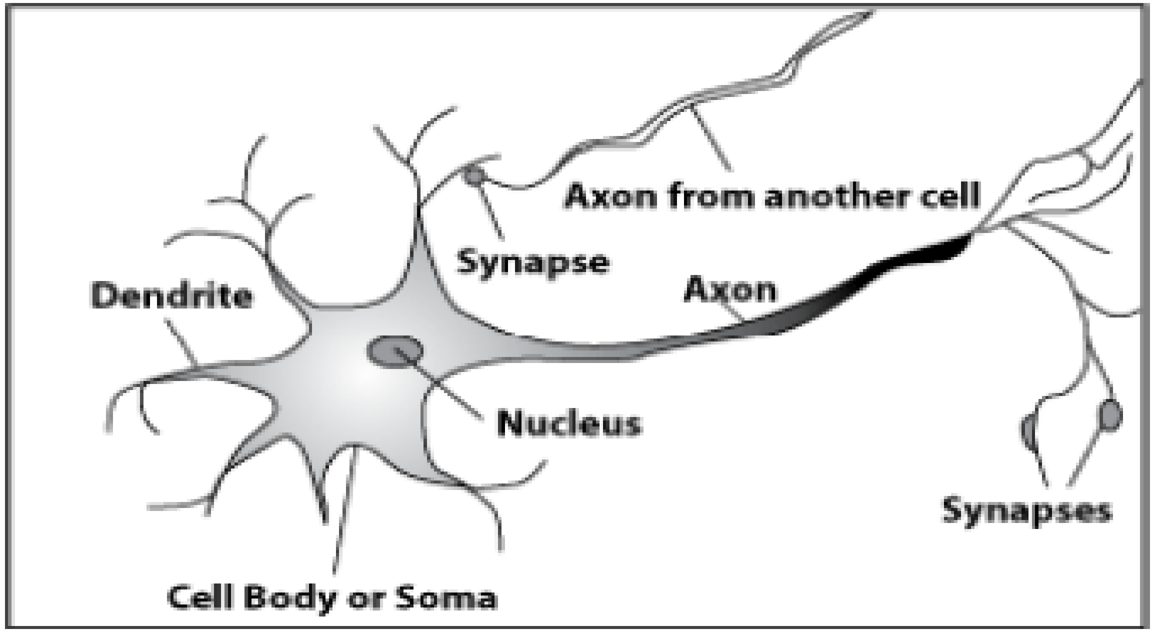
Biological Neural Networks

- In biology, a neuron is a cell that can transmit and process chemical or electrical signals.

- A neuron is connected with other neurons to create a network.

- Tens of billions of interconnected neuron structures – in human brain.

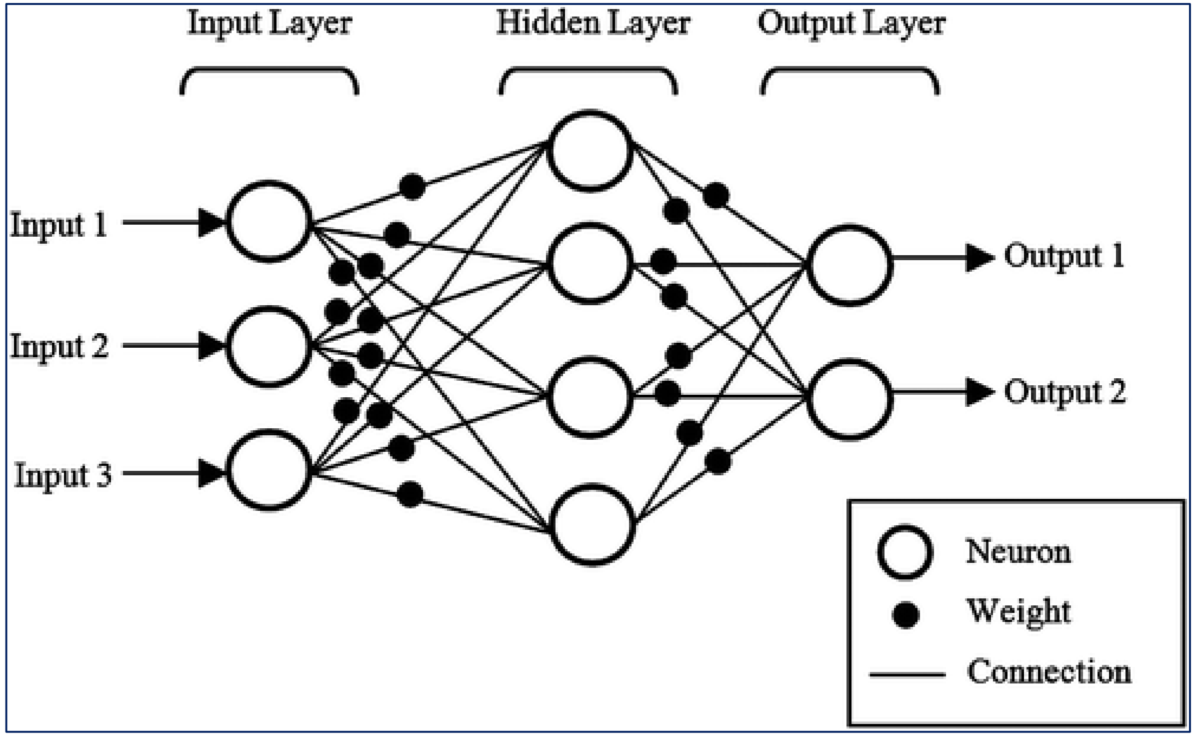
- Every neuron has an
    - Input called the dendrite
    - Cell body called Soma
    - Output called the axon



Artificial Neural Network

Artificial Neural Network

- Pool of simple processing units
- Communication to each other over a large number of weighted connections.



Biological Vs Artificial Neural Networks

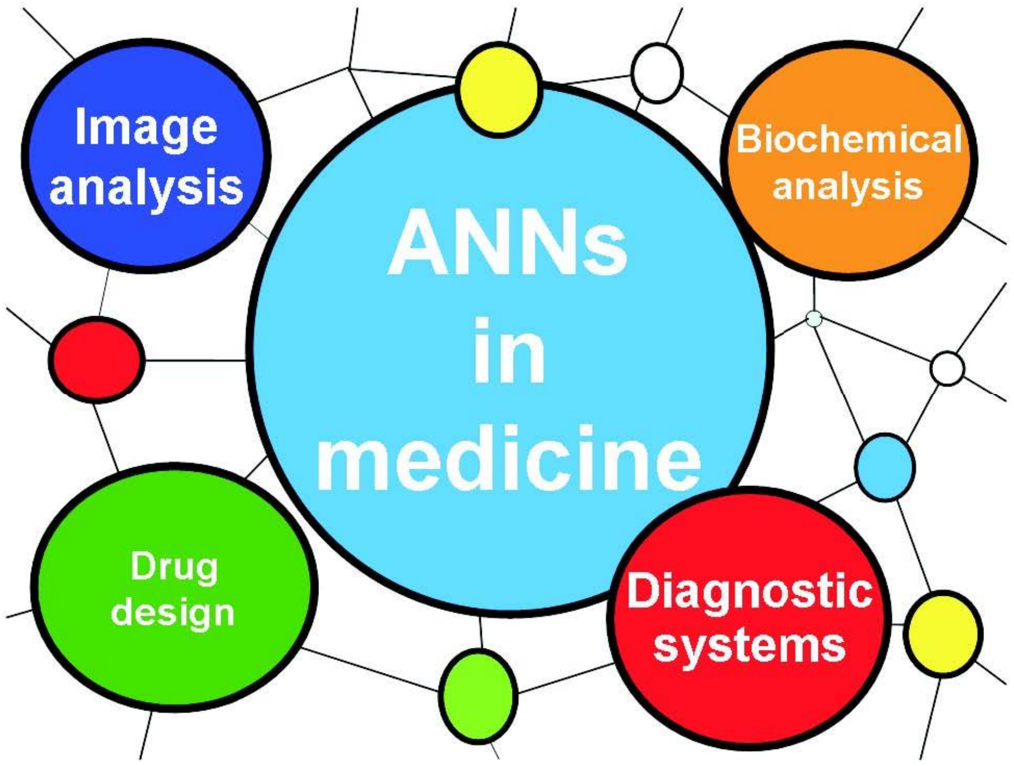
	
Biological neurons or nerve cells	Silicon transistors
200 billion neurons, 32 trillion interconnections.	1 billion bytes RAM, trillion of bytes on disk.
Neuron size: 10-6 m.	Single transistor size: 10-9m.
Energy consumption: 6-10 joules per operation per sec.	Energy consumption: 10-16 joules per operation per second.
Learning capability	Programming capability

Applications of ANN

- Inspired by biological neural networks
  - Numerous advances have been made in developing intelligent systems.

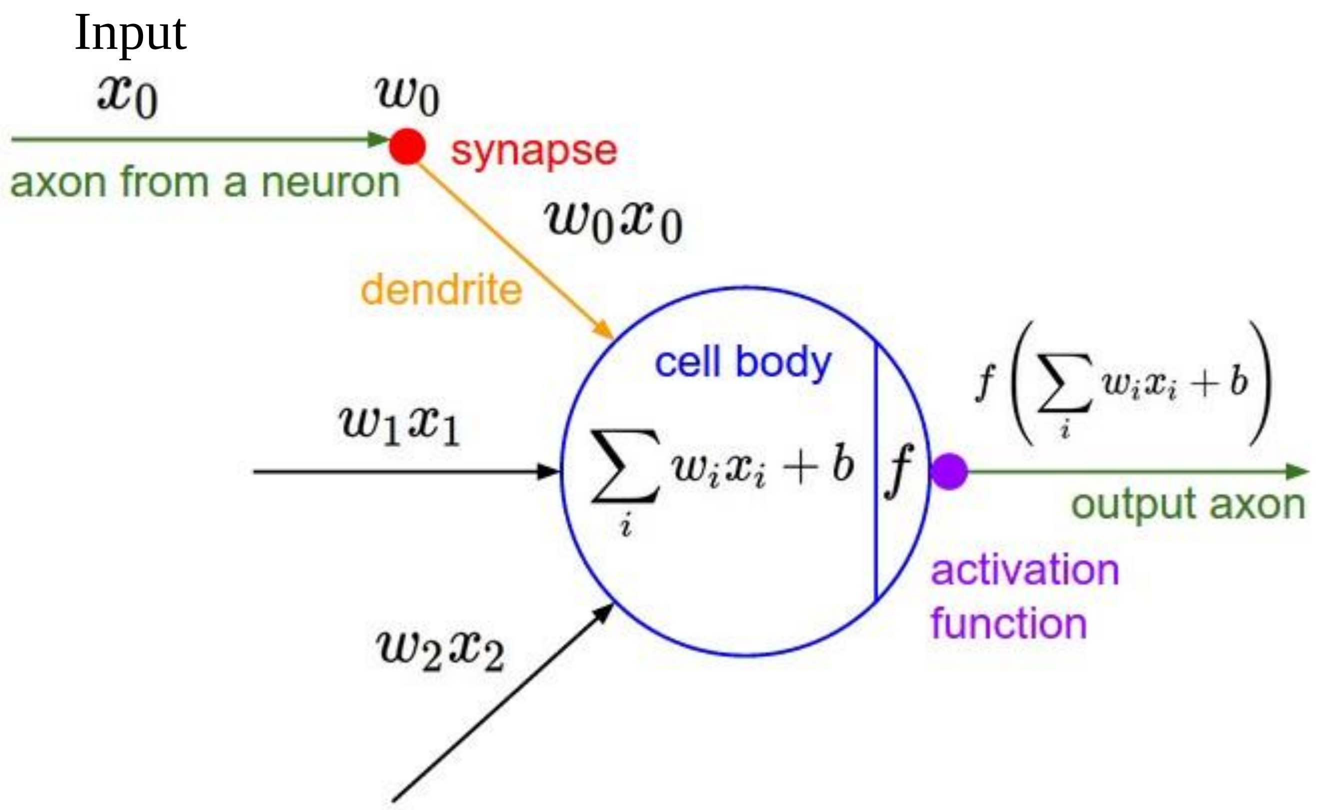
- ANNs developed to solve a variety of problems in
  - pattern recognition
  - prediction
  - optimization
  - associative memory

Example: ANN in medical Applications



Computational Model of ANNs

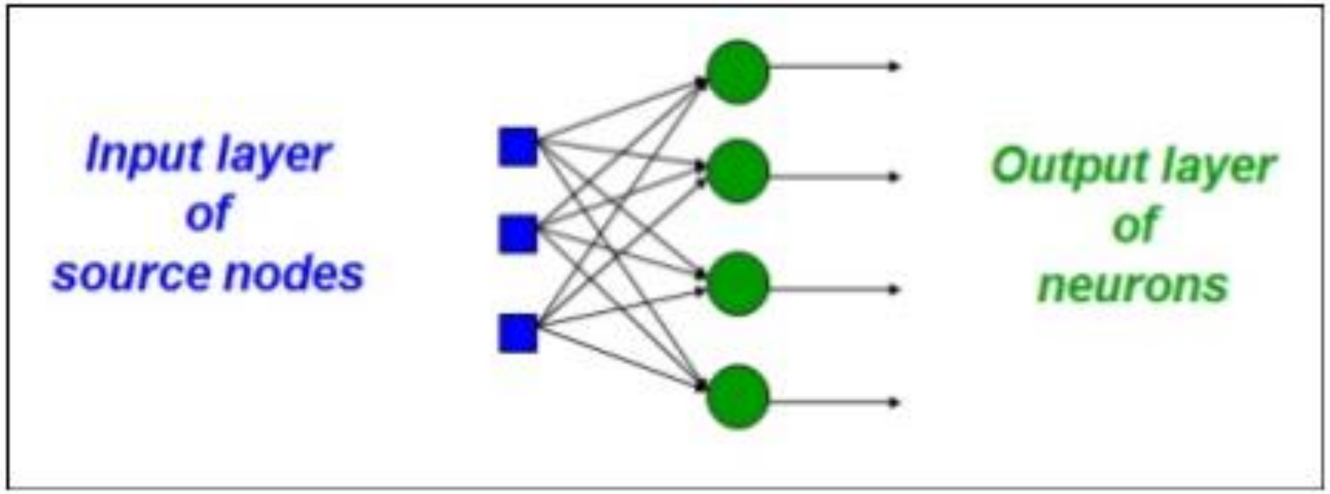
- ANN is an information processing system
  - Consists of many nodes called **neurons** (processing units)
  - Signals are transmitted by connection **links**
  - Links possess an associated **weight**, which is multiplied with input signal (net input)
  - Output is obtained by applying **activations** to net input.



ANN Models

- ANN Models - Classified
  - Single Layer ANN
  - Multi-layer ANN

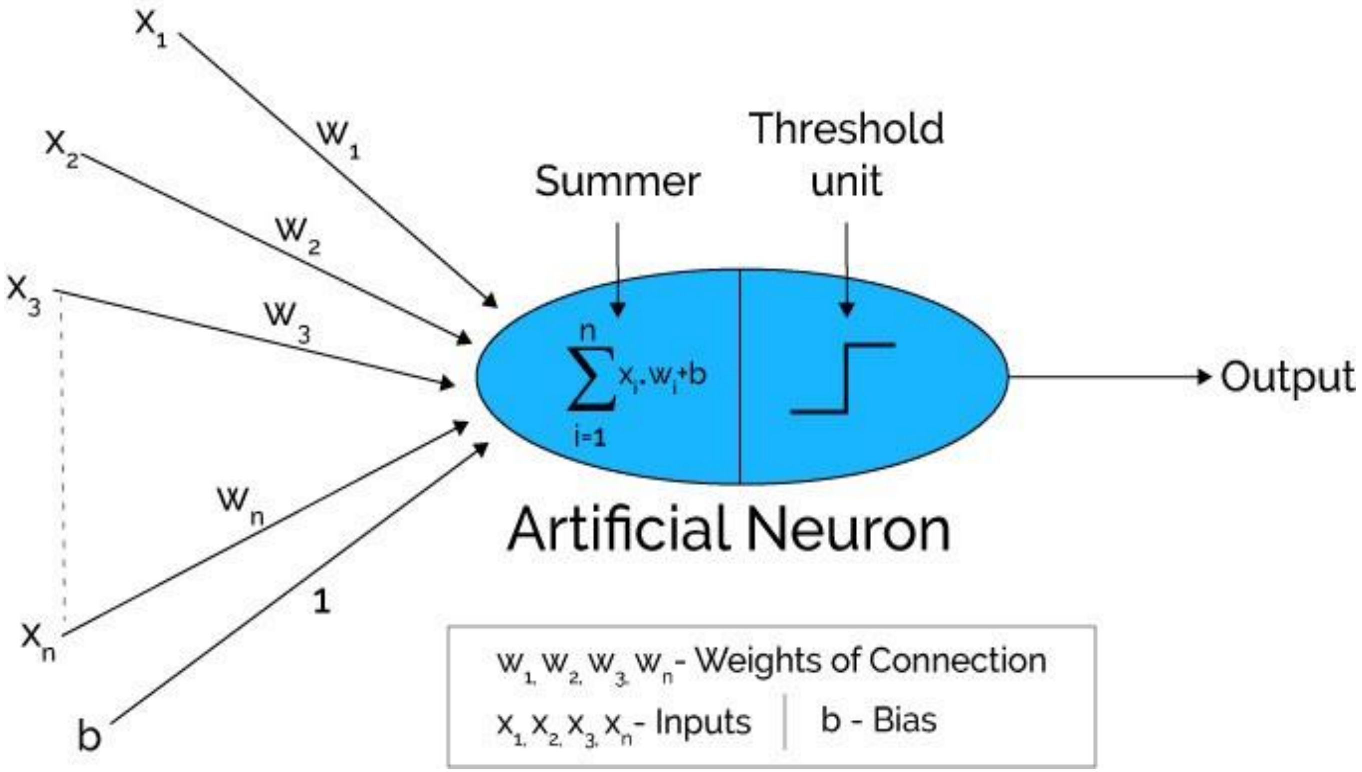
Single Layer ANN



- Only one layer of weighted interconnections
- Weighted input(s) are processed by only one layer and provide output(s)

ANN Models

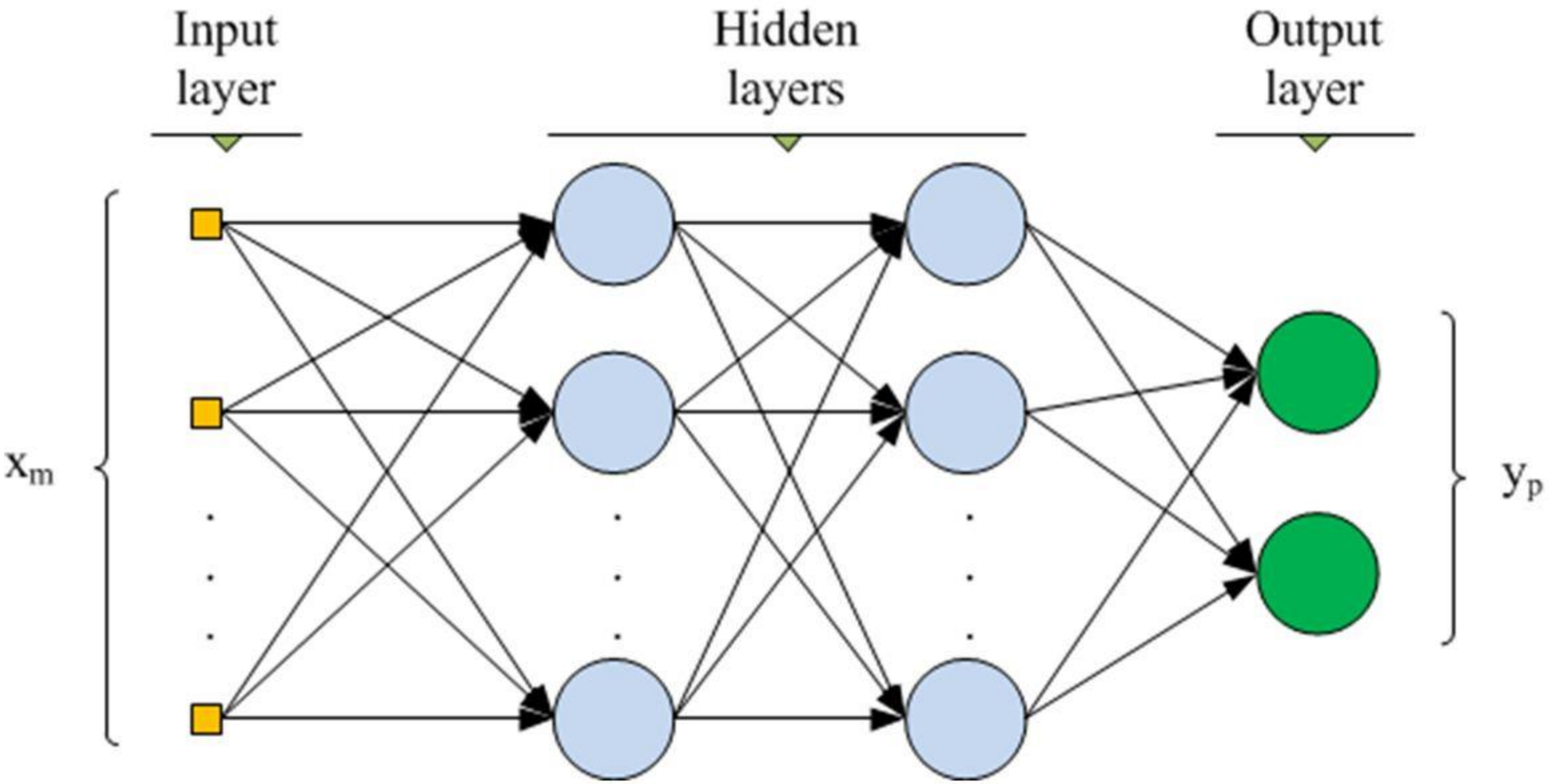
- Single Layer ANN - Example



$X_{1,}$ x_2 … are input
b – bias weight
w_1, w_2 … are interconnecting weights

Multilayer ANN

- Multilayer ANN are called layered networks
  - Input layer
  - Hidden layer(s)
  - Output layer

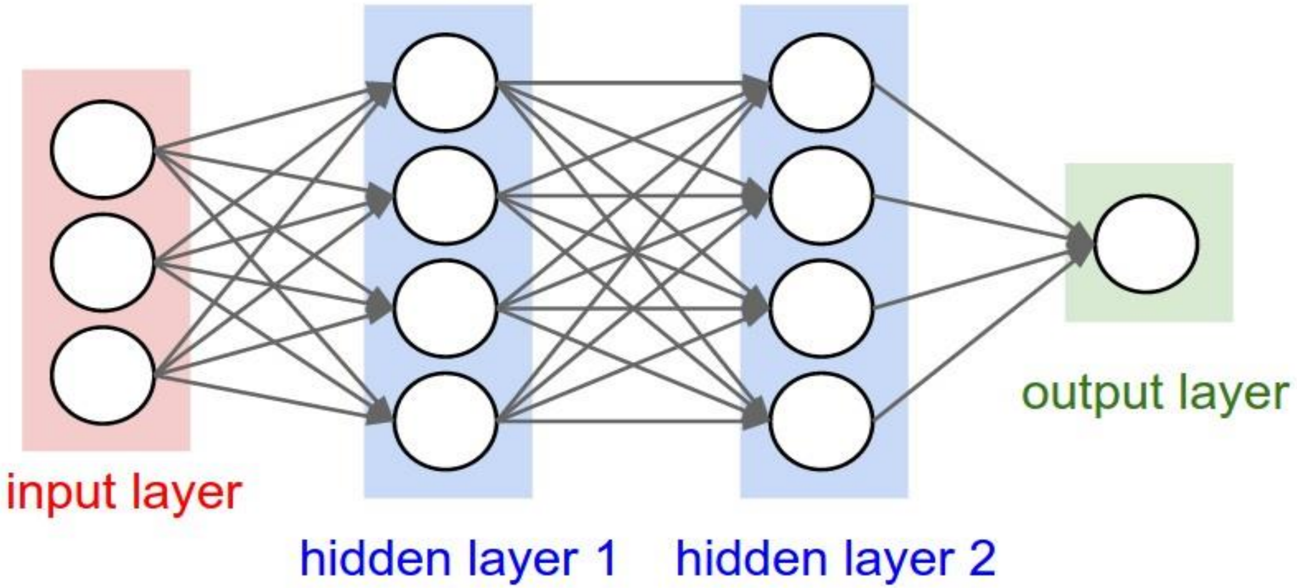


Basic building blocks of ANN

- Network Architecture
- Setting weights
- Activation function

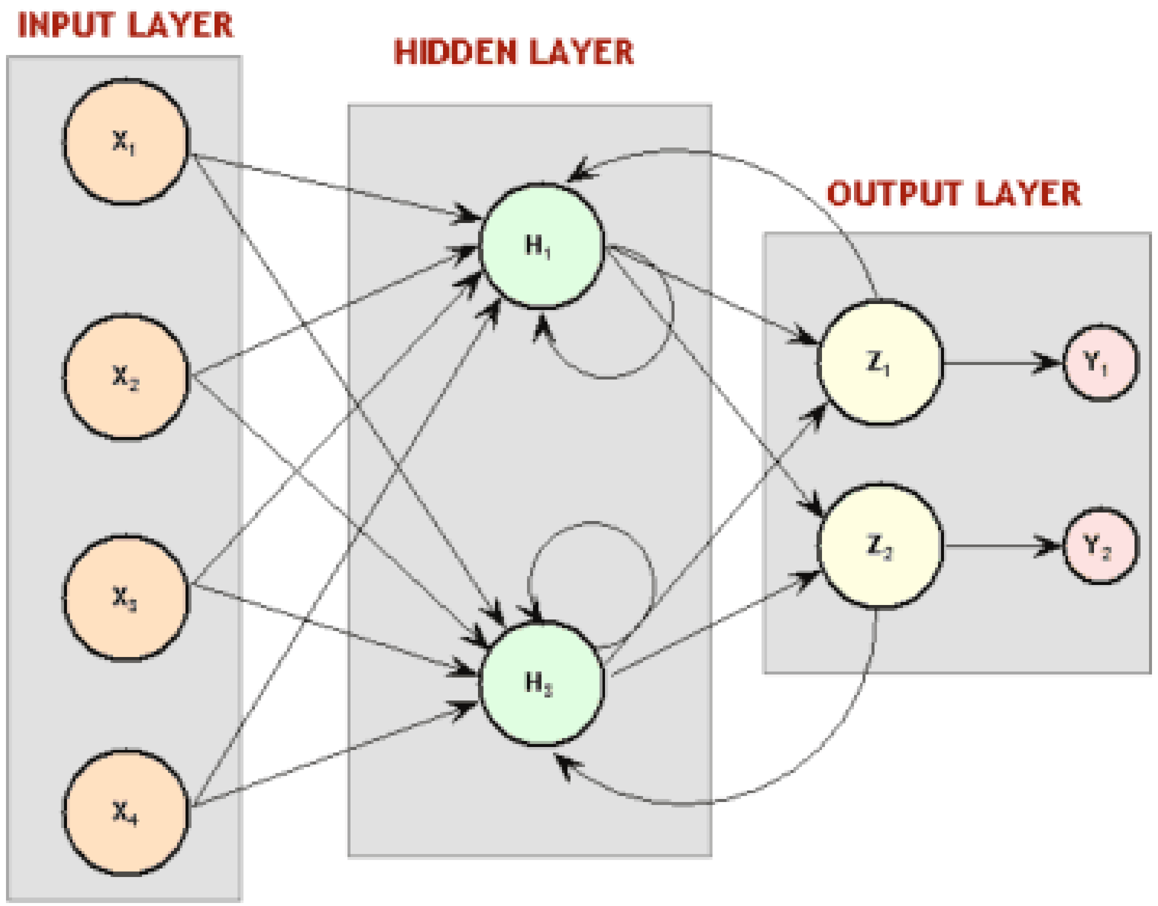
- Network Architecture
  - Arrangement of neurons into layers
  - Connection pattern between neurons

Feed Forward Neural Networks



- The information is propagated from the inputs to the outputs
- Time has no role (NO cycle between outputs and inputs)

Feedback/ Recurrent Neural Networks



- All nodes are connected to all other nodes
- Every node is both input/ output node
- Delays are associated
- Training is more difficult
- Performance may be problematic
  - Stable Outputs may be more difficult

Setting weights

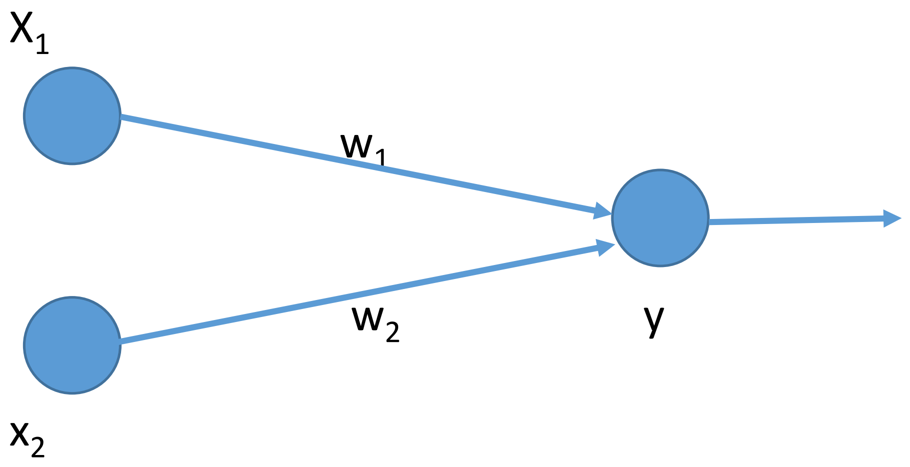
- Setting the values for weights – enable learning/ training

- Training
  - Process of modifying the weights in the network to achieve the expected output

- Learning
  - Internal process when the network is trained

Artificial Neural Network (ANN) Terminologies
1. Weights

- Weight is an information used by the neural net to solve the problem
- Weights can set to zero or can be calculated by some methods



- x_1 – activation of neuron-1 (input signal)
- x_2 – activation of neuron-2 (input signal)
- y – output neuron
- w_1 – weight connecting neuron-1 to output
- w_2 –weight connecting neuron-2 to output

Net input = Net = $x_1 w_1 + x_2 w_2$

Net input = Net $= \sum_{i=1}^{n} x_i w_i$

Artificial Neural Network (ANN) Terminologies

2. Activation Functions/ Transfer Function

- Used to calculate the output response of a neuron.
- Sum of the weighted input signal is applied with an activation to obtain the response.
- For neurons in the same layer same activation functions are used.

- Activation function
  - Linear
  - Non-linear (used in multilayer net)

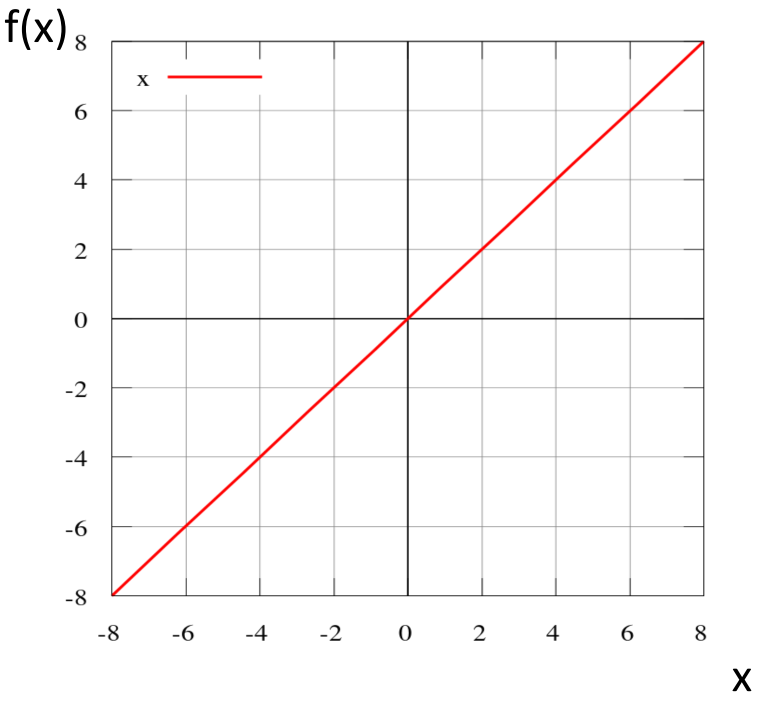
θ

Artificial Neural Network (ANN) Terminologies

2. Activation functions

Identity (Linear) Function:

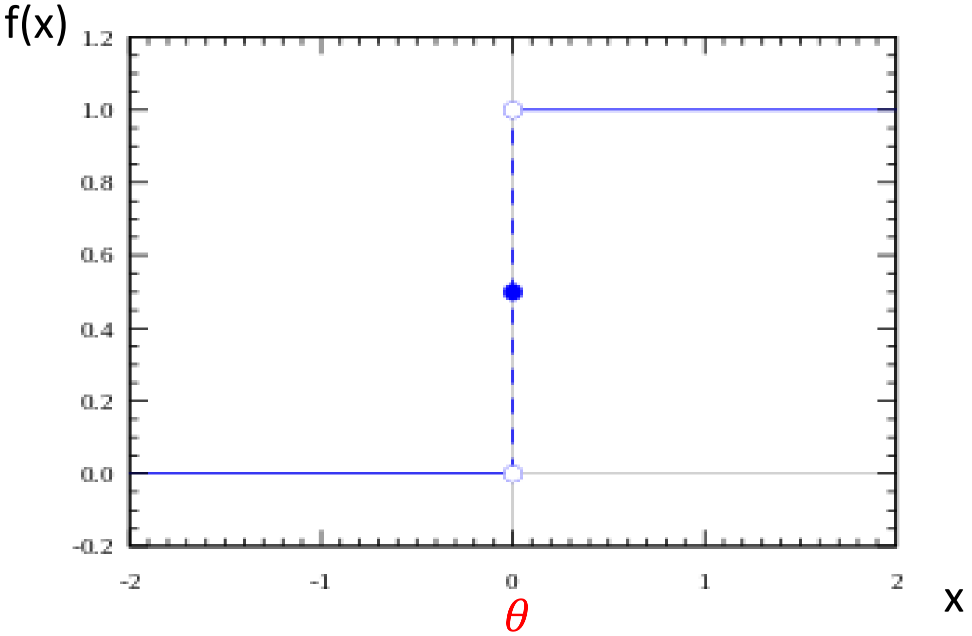
f(x) = x, for all x



Binary Step Function

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

- $where\ \theta\ is\ threshold$
- Single layer nets uses binary step (threshold) function.



Artificial Neural Network (ANN) Terminologies
2. Activation functions

- **Sigmoidal function**
  - S-shaped curves
  - Hyperbolic & logistic functions
  - Used in multi layer nets
  - Example: back propagation net

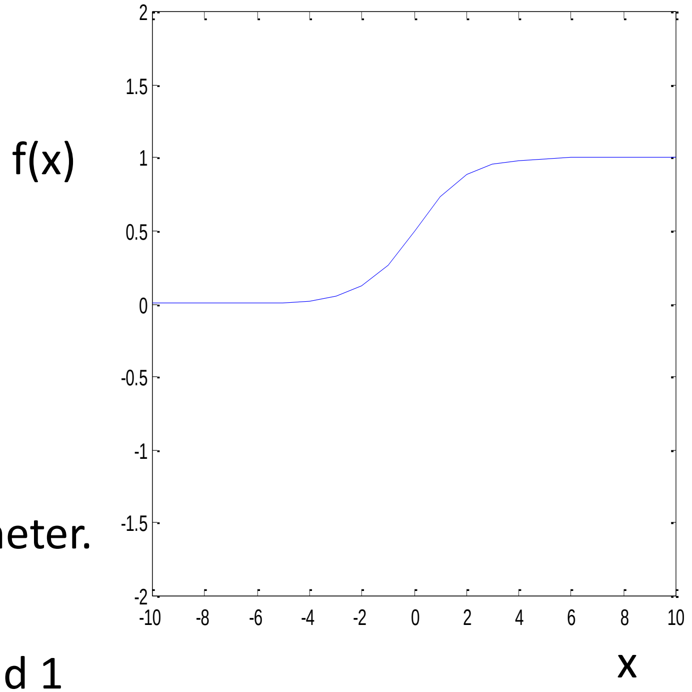
  - Types
    - Binary Sigmoidal Function
    - Bipolar Sigmoidal Function

Binary Sigmoidal (Logistic) Function

It ranges between 0 to 1.

$$f(x) = \frac{1}{1+\exp(-\sigma x)}$$

Where σ – steepness parameter.



- Since range is between 0 and 1
- This is especially used for models where we have to predict probability

Artificial Neural Network (ANN) Terminologies

2. Activation functions

- **Bipolar Sigmoidal Function**
  - Range: +1 and -1
  - Called tanh/ hyperbolic tangent function

$$b(x) = 2f(x) - 1$$

$$b(x) = 2 \times \frac{1}{1+\exp(-\sigma x)} - 1$$

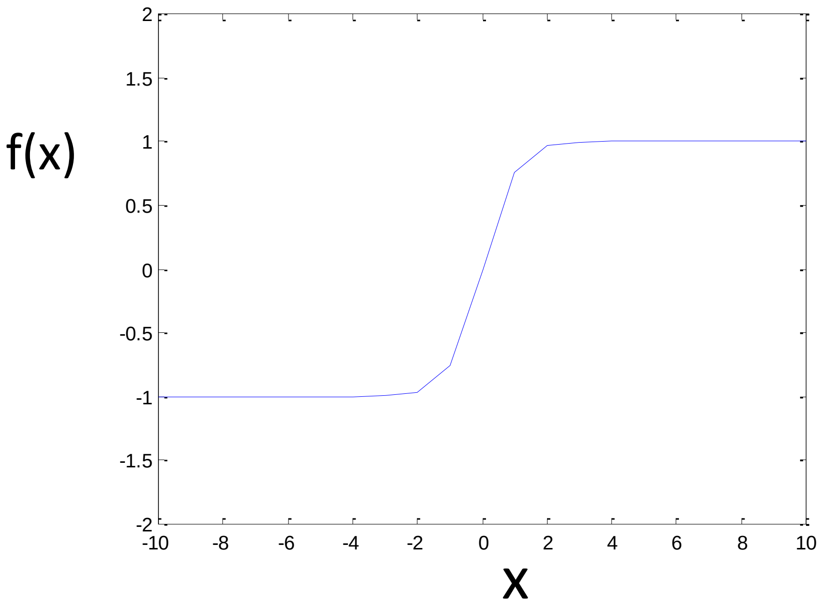
$$b(x) = \frac{2 - 1 - \exp(-\sigma x)}{1+\exp(-\sigma x)}$$

$$b(x) = \frac{1-\exp(-\sigma x)}{1+\exp(-\sigma x)}$$

Where σ – steepness parameter.

Bipolar Activation Function

Used for classification
Used in feed-forward nets.



Artificial Neural Network (ANN) Terminologies
2. Activation functions

Rectified linear unit (ReLU):

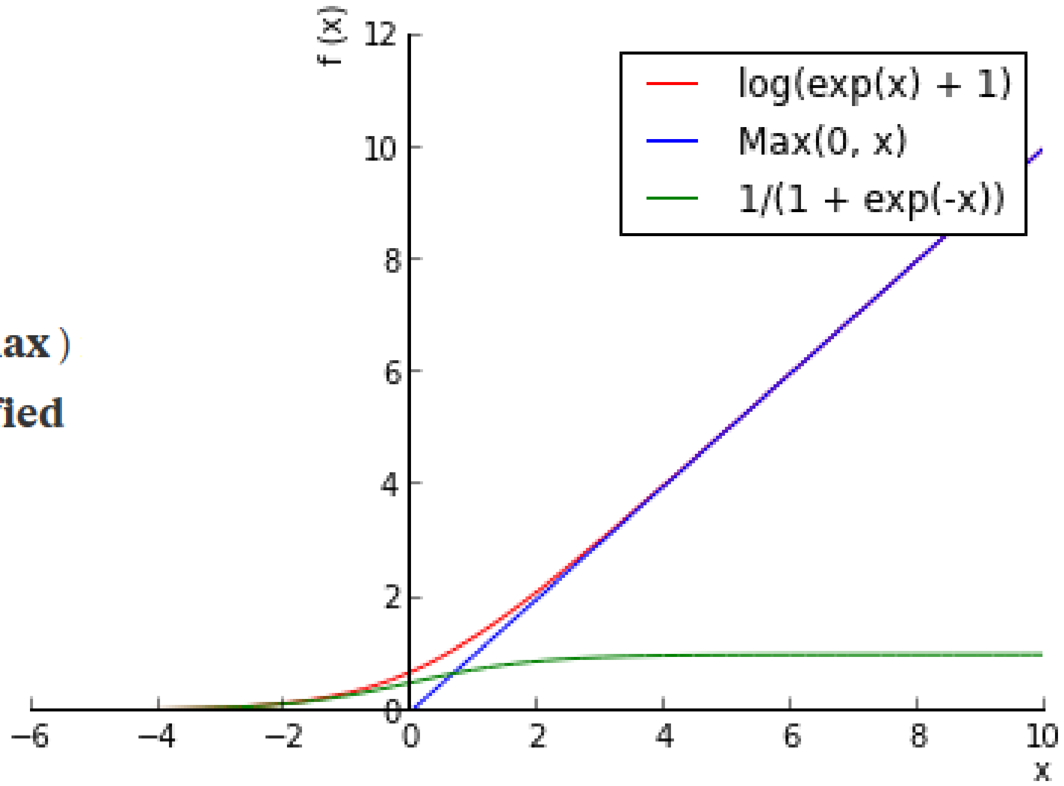
$f(x) = \sum_{i=1}^{\inf} \sigma(x - i + 0.5) \approx log(1 + e^x)$

we refer

- $\sum_{i=1}^{\inf} \sigma(x - i + 0.5)$ as **stepped sigmoid**
- $log(1 + e^x)$ as **softplus function**

The softplus function can be approximated by **max function** (**or hard max**) $max(0, x + N(0,1))$. The max function is commonly known as **Rectified Linear Function (ReL).**

$\boxed{max\ (0, x)}$



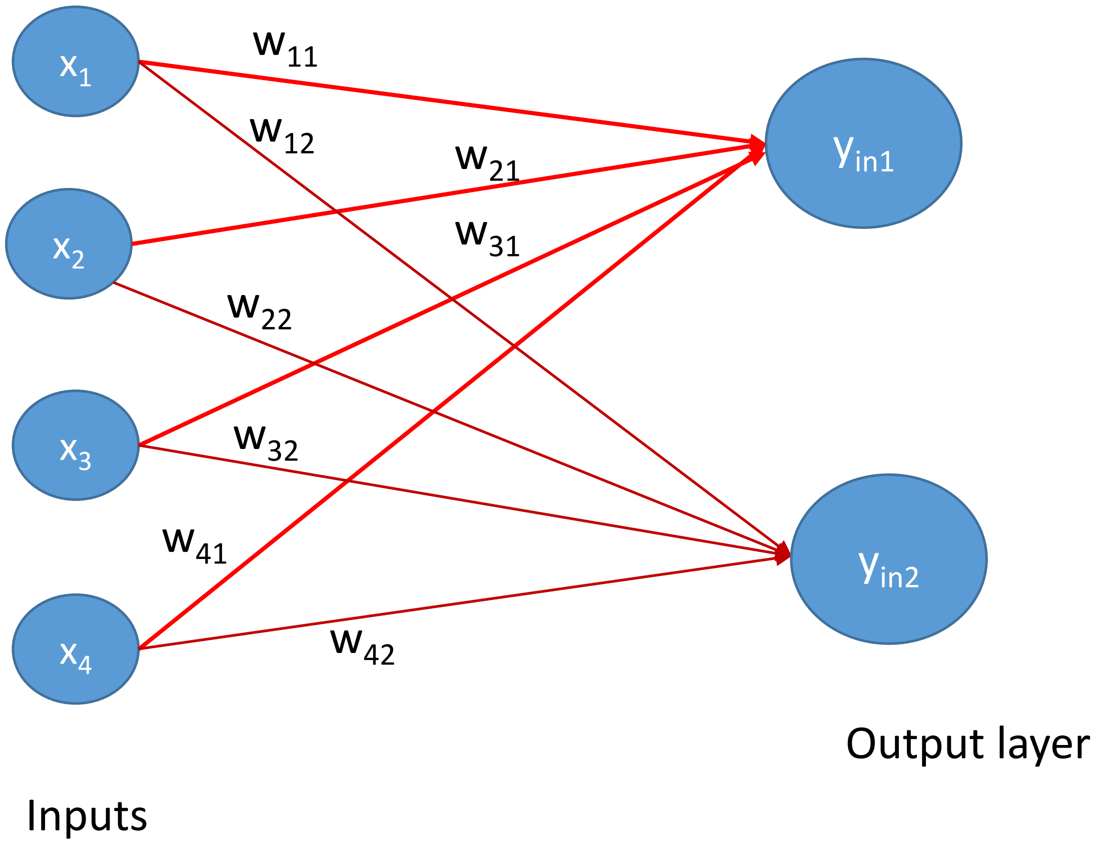
Artificial Neural Network (ANN) Terminologies

3. Calculation of Net Input using matrix multiplication method

- If the weights are given as W = (w_{ij}) in a matrix form

- The net input to output $y_{inj} = x_i * w_{ij}$

$$y_{inj} = \sum_{i=1}^{n} x_i \, w_{ij}$$

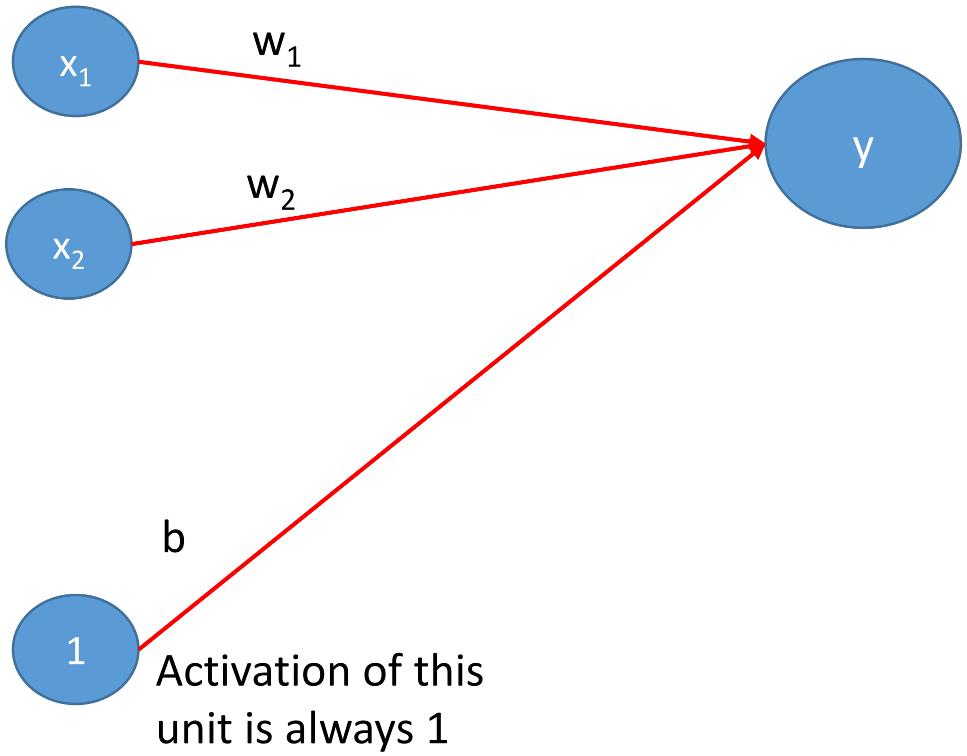


Artificial Neural Network (ANN) Terminologies

4. Bias

- Weight on a connection from a unit whose activation is always 1.
- Increasing bias increase net input.

$$\mathrm{Net} = \mathrm{b} + \sum_{i=1}^{n} \mathrm{x_i}\ \mathrm{w_i}$$



Artificial Neural Network (ANN) Terminologies
5. Threshold

- Used in calculating the activations of the given net
- Based on the threshold, output is calculated.
- Activation function is based on the value of θ

$$y = f(Net) = \begin{cases} +1;\ if\ Net \geq \theta \\ -1; if\ Net < \theta \end{cases}$$

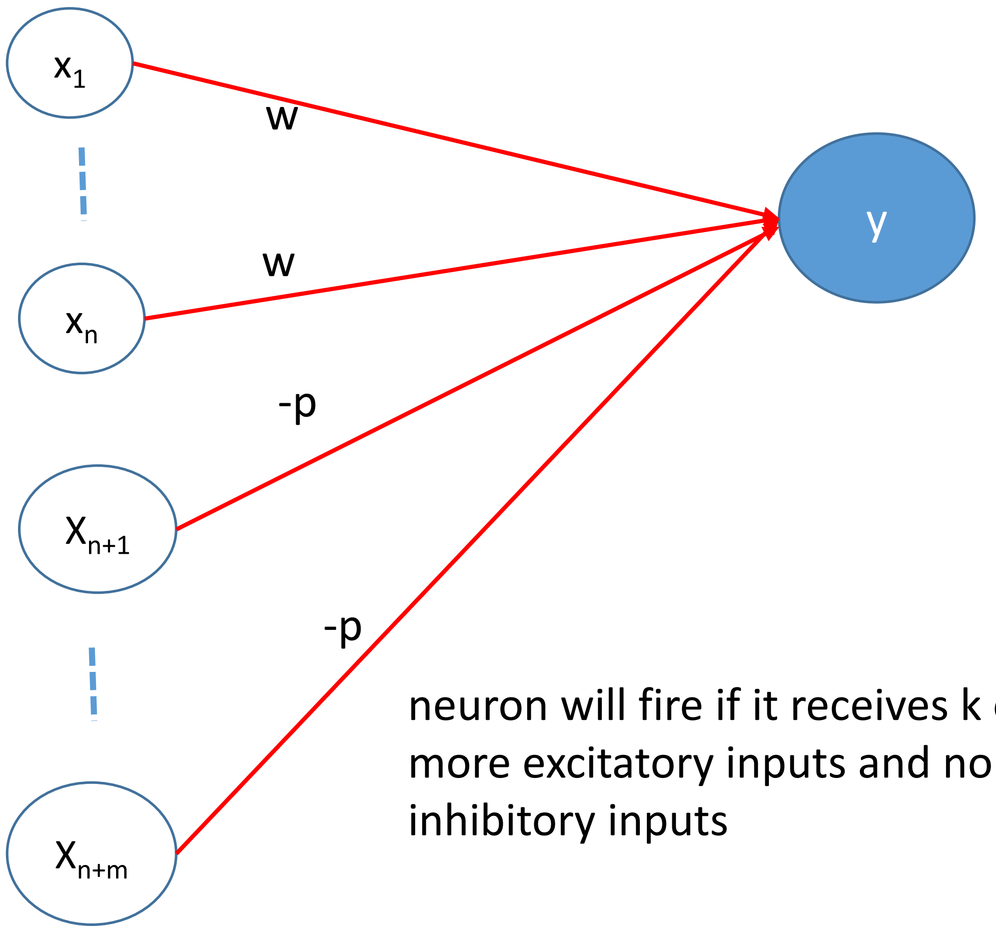
$where\ \theta\ and\ \theta_j\ are\ thresholds$

McCulloch-Pitts Neuron Model

- Here, y is McCulloch-Pitts neuron
- Receives signal from any number of neurons
- Connection weights from x_1 .. x_n are excitatory, denoted by w.
- Connection weights from x_{n+1} .. X_{n+m} are inhibitory, denoted by –p.
- McCulloch-Pitts neuron y has the activation function

$$\boldsymbol{f(y_{in}) = \begin{cases} 1; if\ f(y_{in}) \geq \theta \\ 0; if\ f(y_{in}) < \theta \end{cases}}$$

- Where, θ – threshold and y_{in} - net input to y



neuron will fire if it receives k or more excitatory inputs and no inhibitory inputs

Examples

- Generate the output of logic AND function by McCulloch-Pitts neuron model.

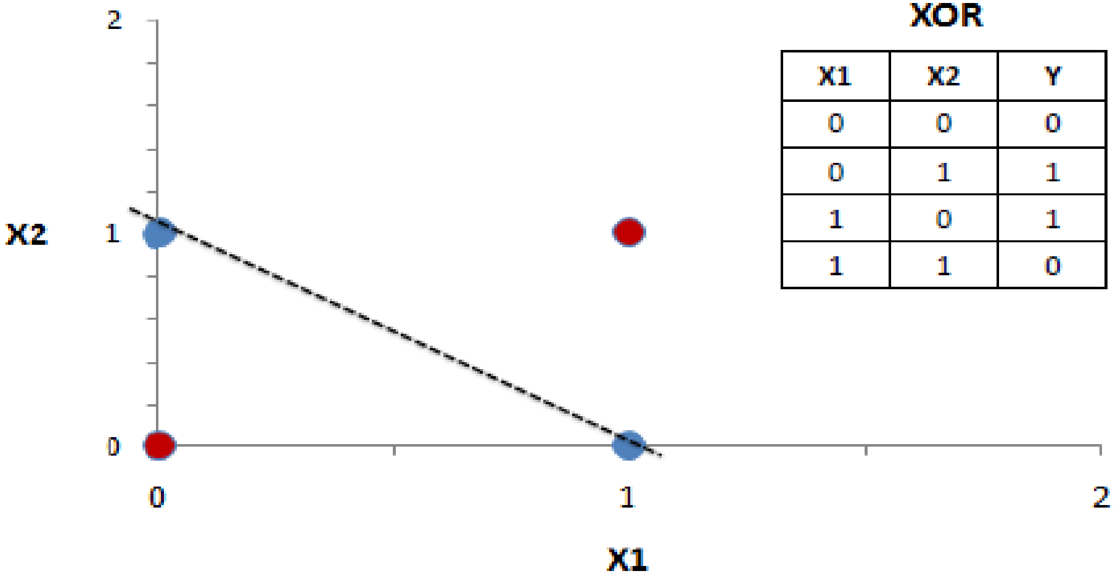
- Generate the output of logic OR function by McCulloch-Pitts neuron model.

- Generate the output of logic NOT function by McCulloch-Pitts neuron model.

- Generate the output of logic XOR function by McCulloch-Pitts neuron model.

Perceptron Networks

- Frank Rosenblatt
- Minsky and Papert - limitations
- Learning rule:
  - Iterative weight adjustment (powerful technique)
- Training in perceptron
  - Continue until no error (minimum) occurs.
- Perceptron Net
  - Solve – classification problem



XOR

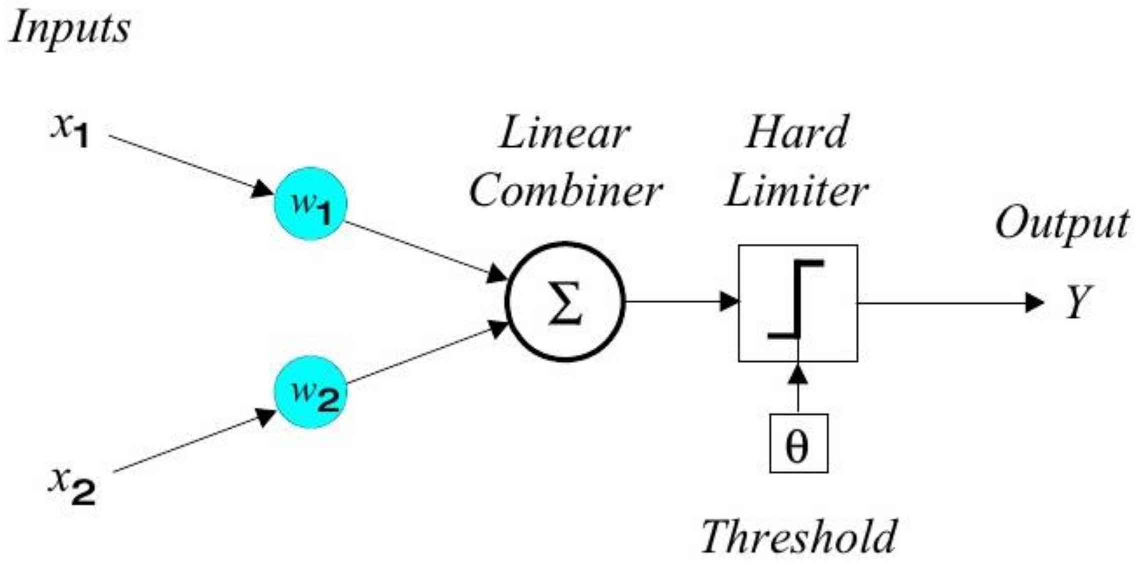
X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

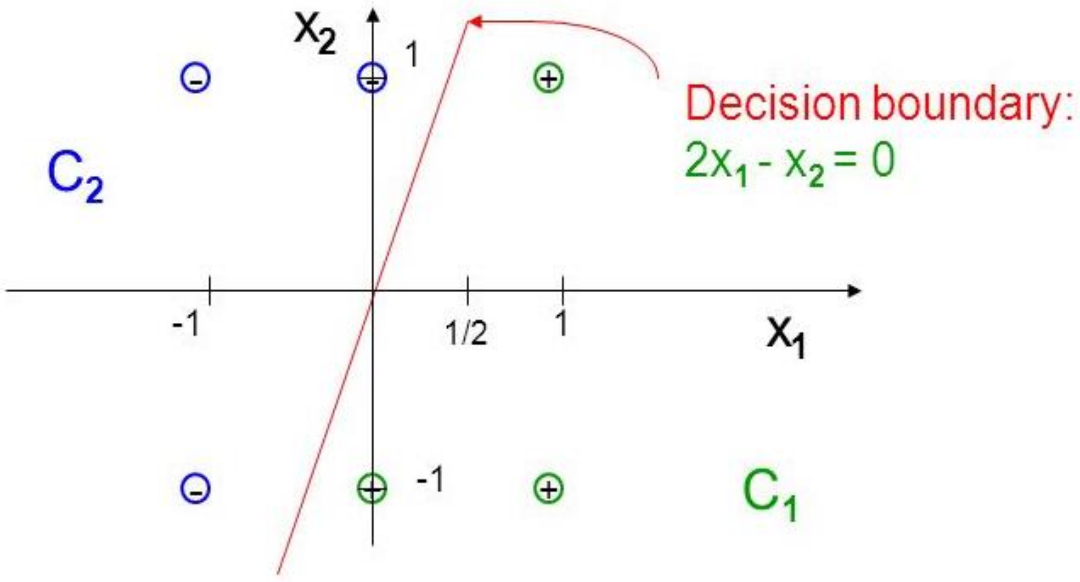
Perceptron Networks

- Types of perceptron:
  - Single Layer perceptron
  - Multilayer perceptron

Single Layer Perceptron

- Used for classification of patterns that are linearly separable.
- It consists of single neuron, that adjust the weights and bias.
- Rosenblatt found
  - If the patterns used to train the perceptron are drawn from the linearly separable class, the perceptron algorithm converges
  - Positions the decision surface in the form of a hyper plane between two classes.





Single Layer Perceptron

Algorithm

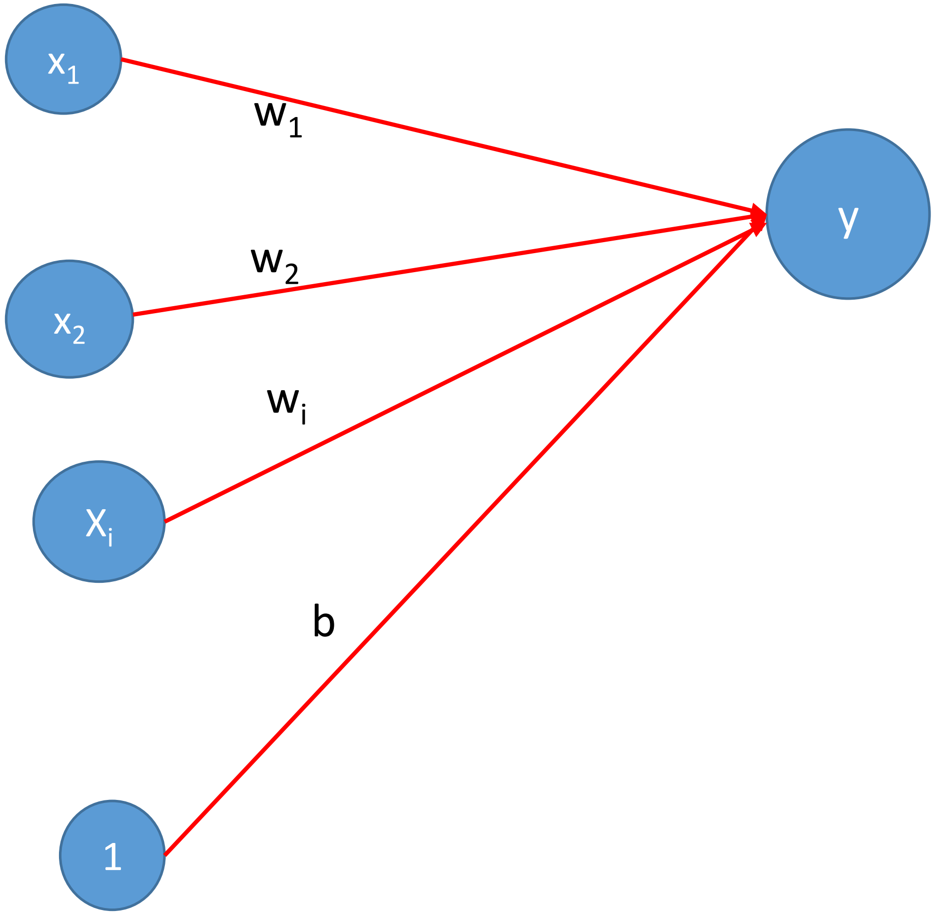
1. Initialize weights and bias (set to ZERO) and set the learning rate α (0 to 1)

2. While stopping condition is not TRUE, repeat steps 3 to 7

3. For each training pair S: t do steps 4 to 6

4. Input $x_i = s_i$ for all i = 1 to n

Architecture



Single Layer Perceptron Architecture

5. Compute output response

$$y_{in} = b + \sum_{i=1}^{n} x_i \, wi$$

Activation function

$$Y = f(y_{in}) = \begin{cases} +1 & if\ y_{in} > \theta \\ 0 & if\ -\theta \le y_{in} \le \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

6. If the output response is not equal to target, then update the weights and bias

If $t \neq y$

$$w_{i(new)} = w_{i(old)} + \alpha t x_i$$
$$b_{(new)} = b_{(old)} + \alpha t$$

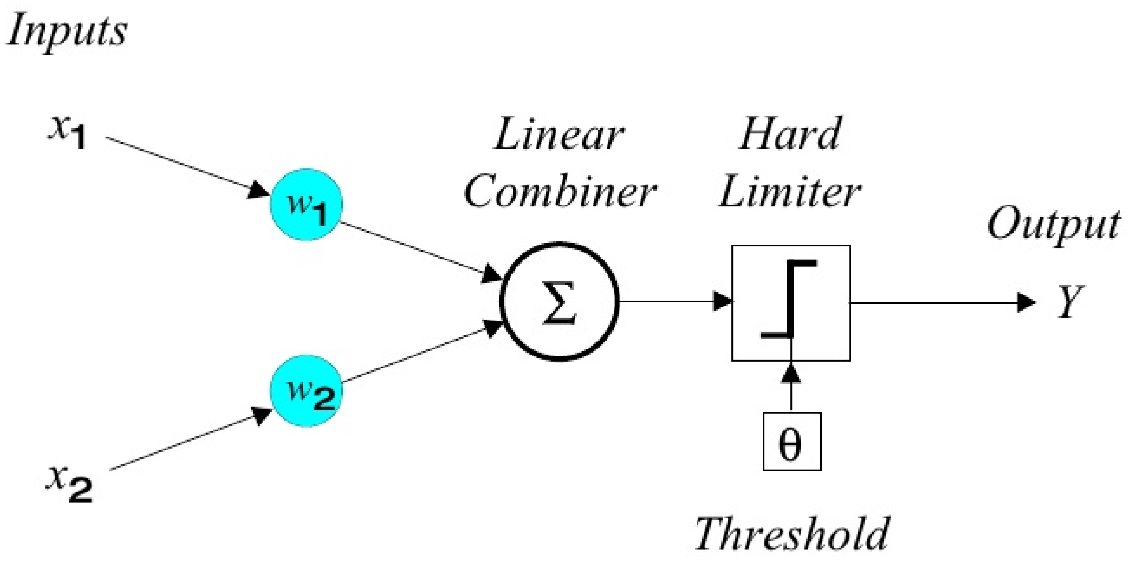
else

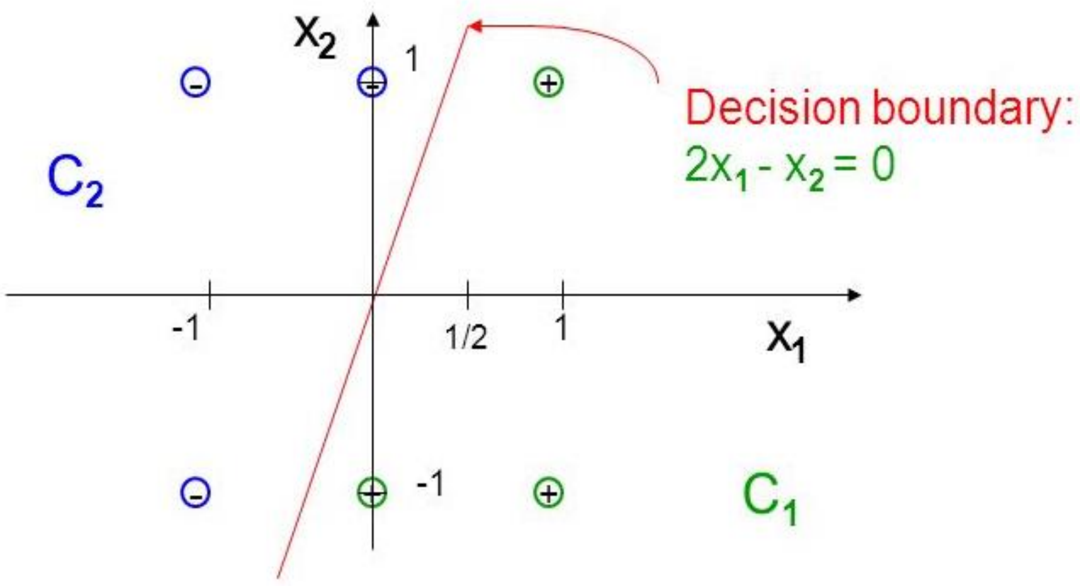
$$w_{i(new)} = w_{i(old)}$$
$$b_{(new)} = b_{(old)}$$

7. Test for stopping condition

Single Layer Perceptron Limitation

- This model is limited to perform the classification with only two classes.





Gradient Descent

- It is a technique that update the weights in every iteration to minimize the error of a model on our training data.

  - Input: Training instances $<0, 0>$, $<0, 1>$, $<1, 0>$, $<1, 1>$ one at a time.
  - Model make a prediction for the training instances
  - Calculate the error = $\alpha t x_i$
  - Update weights to reduce the error for next prediction

$$w_{i(new)} = w_{i(old)} + \alpha t x_i$$

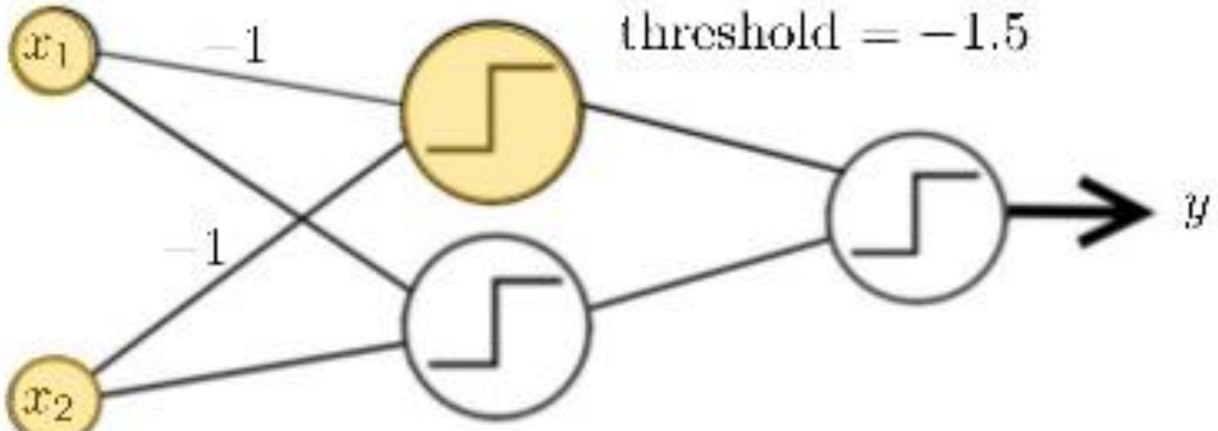
Example

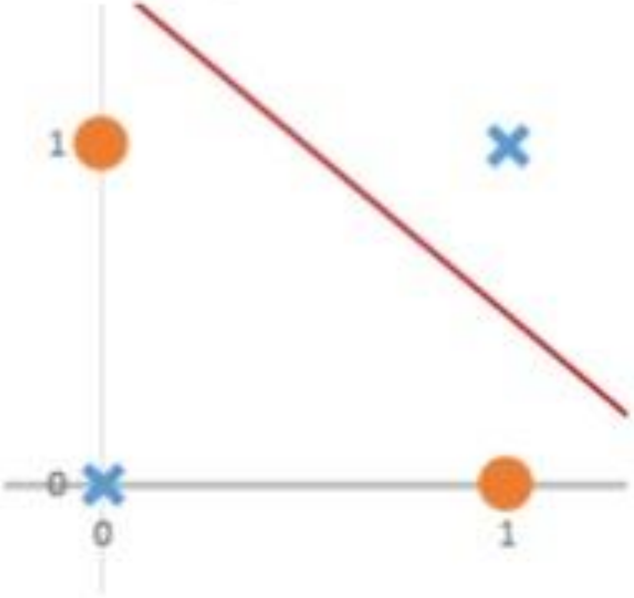
- Develop a perceptron for AND function with bipolar inputs and targets.
- Develop a perceptron for OR function with bipolar inputs and targets.

XOR Problem

Training Data

x_1	x_2	t
0	0	0
1	0	1
0	1	1
1	1	0





Multiple layers of perceptrons solve the XOR problem, but Rosenblatt did not have an learning algorithm to set the weights

- Single layer NN
  - Linear classification
  - Linearly separable
- Multilayer NN
  - Non-linear classification
  - All Boolean fun can be represented by the two layer (single hidden layer)
  - Any function can be classifiable with minimum error using Artificial Neural Network with two hidden layers.

Multilayer Networks

- Feed forward Networks
  - Output is calculated for every input to the network
  - No feedback

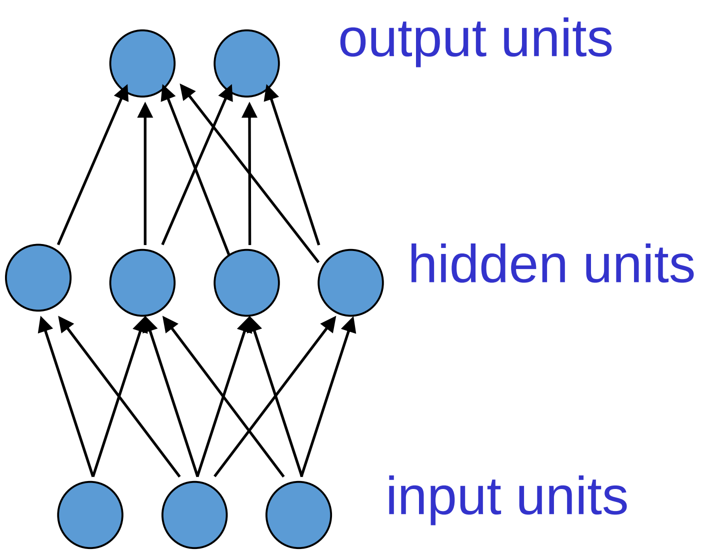
  - Example: Back Propagation Network

- Back Propagation Network (BPN)
  - Multilayer ANN
  - Using gradient descent based delta learning rule known as back propagation (for errors)
  - Efficiently changing the weights with **differentiable activation function** units to learn a training set of input-output examples.

Back Propagation Network (BPN)

Architecture

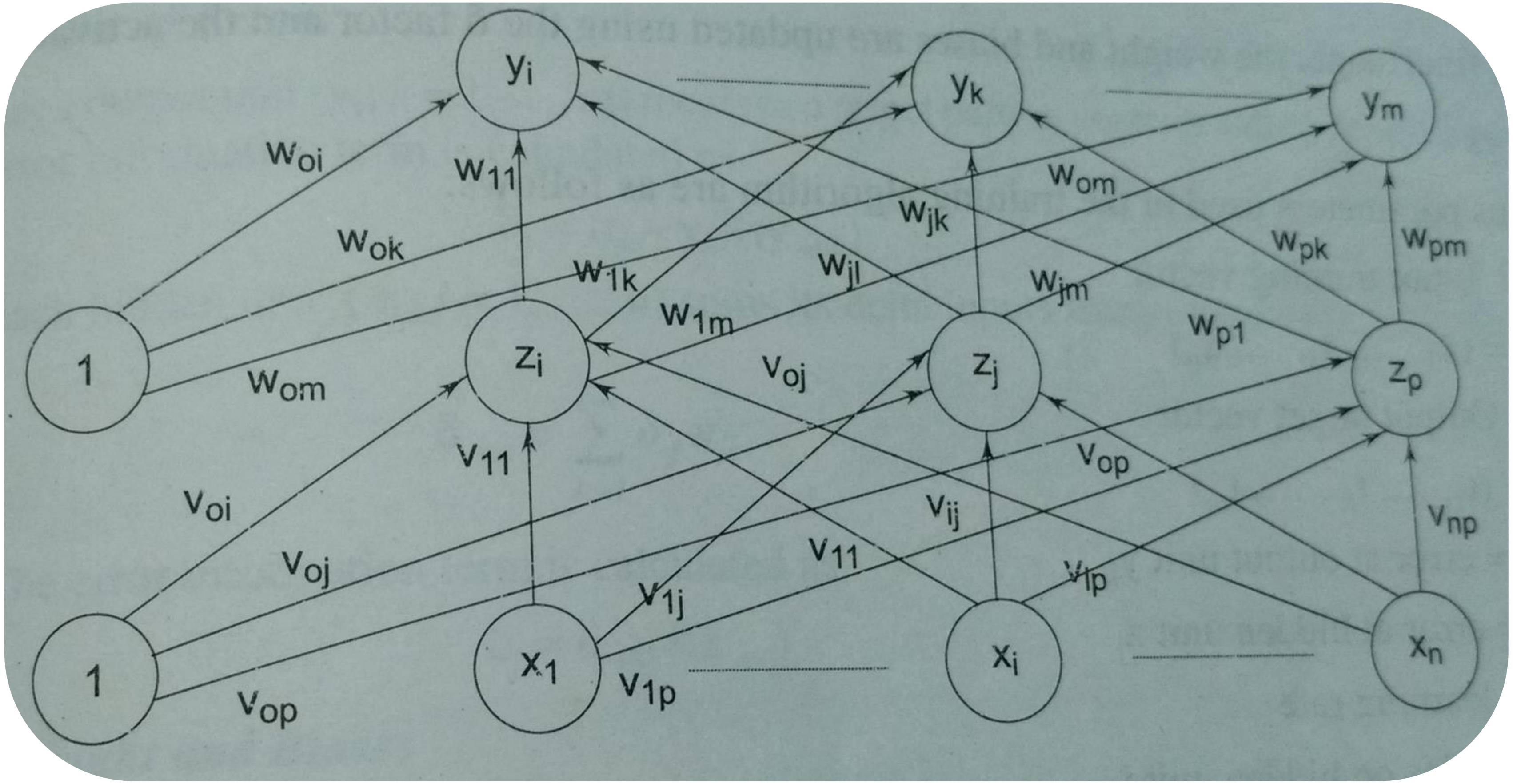
- These are the commonest type of neural network in practical applications.
    - The first layer is the input and the last layer is the output.
    - If there is more than one hidden layer, we call them “**deep**” neural networks.
- They compute a series of transformations that change the similarities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



During back propagation learning
- Signals sent in reverse direction also.

Increase in number of hidden layer
- Minimize error
- Increase computational complexity
- Increase time taken for convergence.

Back Propagation Network (BPN)



Parameters

- The various parameters used in the training algorithm are

x: Input training vector

$x = (x_1, \ldots, x_i, \ldots, x_n)$

t: Output target vector

$t = (t_1, \ldots, t_k, \ldots, t_m)$

δ_k = error at output unit y_k

δ_j = error at hidden unit z_j

α = learning rate

V_{oj} = bias on hidden unit j

z_j = hidden unit j

w_{ok} = bias on output unit k

y_k = output unit k.

Training Algorithm

Step 1: Initialize weight to small random values.

Step 2: While stopping condition is false, do steps 3 – 10

Step 3: For each training pair do steps 4 - 9

Training Algorithm

Feed Forward

Step 4: Each input unit receives the input signal x_i and transmits this signals to all units in the layer above i.e. hidden units

Step 5: Each hidden unit (z_j, $j = 1, \ldots, p$) sums its weighted input signals

$$z_{-inj} = v_{oj} + \sum_{i=1}^{n} x_i v_{ij}$$

applying activation function

$$Z_j = f(z_{inj})$$

and sends this signal to all units in the layer above i.e output units.

Training Algorithm

Step 6: Each output unit (y_k, k=1, …, m) sums its weighted input signals

$$y_{-ink} = w_{ok} + \sum_{j=1}^{p} z_j w_{jk}$$

and applies its activation function to calculate the output signals.

$$Y_k = f(y_{-ink})$$

Training Algorithm

Back Propagation of Errors

Step 7: Each output unit (y_k, $k = 1, \ldots, m$) receives a target pattern corresponding to an input pattern, error information term is calculated as

$$\delta_k = (t_k - y_k) f(y_{-ink})$$

Step 8: Each hidden unit (z_j, $j = 1, \ldots, n$) sums its delta inputs from units in the layer above

$$\delta_{-inj} = \sum_{k=1}^{m} \delta_k w_{jk}$$

The error information term is calculated as

$$\delta_j = \delta_{-inj} f(z_{-inj})$$

Training Algorithm

Updation of weights and biases

Step 9: Each output unit (y_k, $k = 1, \ldots, m$) updates its bias and weights ($j = 0, \ldots, p$)

The weight correction term is given by

$$\Delta W_{jk} = \alpha \delta_k z_j$$

and the bias correction term is given by

$$\Delta W_{ok} = \alpha \delta_k$$

Therefore, $W_{jk}(\text{new}) = W_{jk}(\text{old}) + \Delta W_{jk}$, $\quad W_{ok}(\text{new}) = W_{ok}(\text{old}) + \Delta W_{ok}$

Each hidden unit (z_j, $j = 1, \ldots, p$) updates its bias and weights ($i = 0, \ldots n$)

The weight correction term

$$\Delta V_{ij} = \alpha \delta_j x_i$$

The bias correction term

$$\Delta V_{oj} = \alpha \delta_j$$

Therefore, $V_{ij}(\text{new}) = V_{ij}(\text{old}) + \Delta V_{ij}$, $\quad V_{oj}(\text{new}) = V_{oj}(\text{old}) + \Delta V_{oj}$

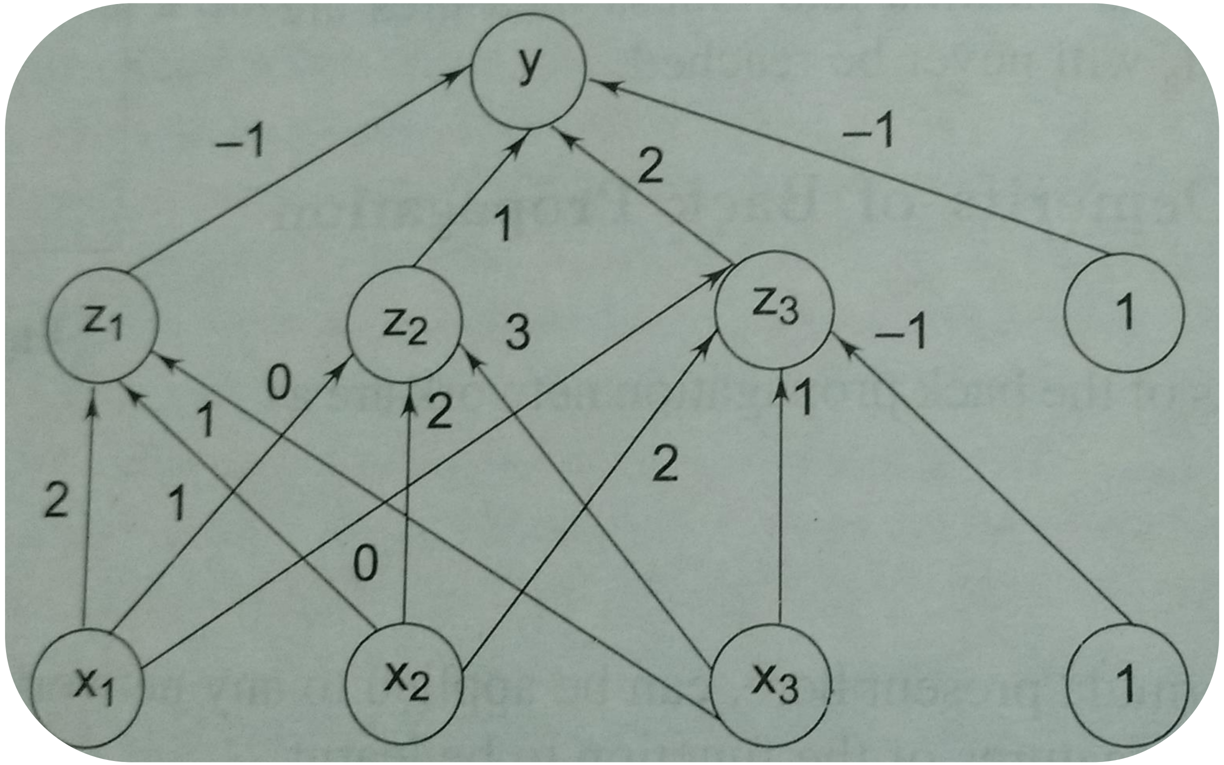
Training Algorithm

Step 10: Test for stopping condition:

The stopping condition may be the minimization of errors, number of epochs etc.

Example

- Find the new weights when the network given in the figure is presented the input pattern [0.6 0.8 0] and the target output is 0.9. use the learning rate α = 0.3 and use binary sigmoid activation function.



Thank you