Perceptrons

CAI 5107: Machine Learning

Instructor: Anowarul Kabir

Email: akabir@usf.edu

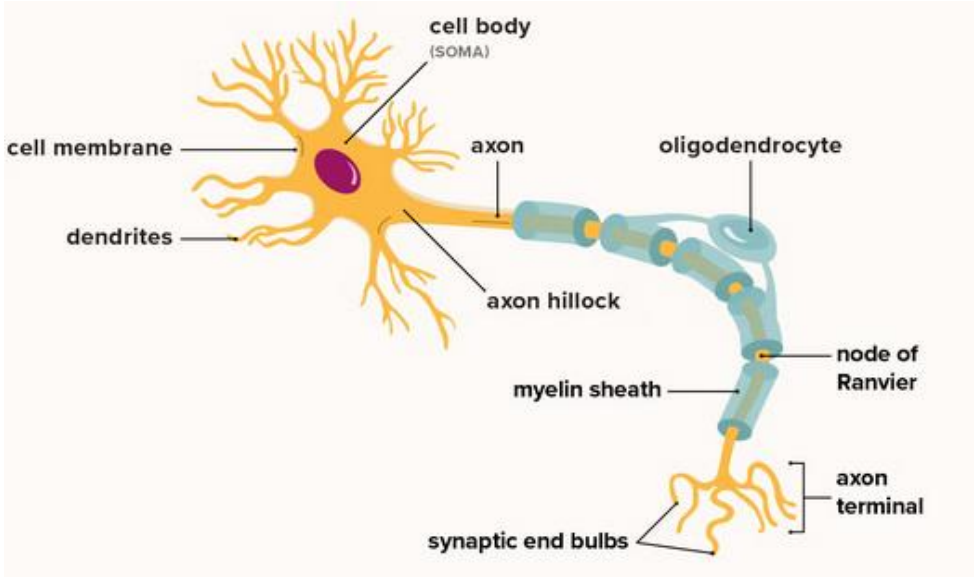
Fall 2025

Outline

- Perceptrons
- Feedforward neural networks (FCNNs)
- Backpropagation

Neural Network (NN)

- NN is a mathematical function
  - Analogically, can be thought of as a neuron



- Benefits
  - Successfully applied in variety of domains
  - Can model complex and nonlinear problems
  - Machine learning problems can be solved with sufficient amount of data
  - Inference is fast

Image source: https://www.healthline.com/health/neurons

A simple architecture

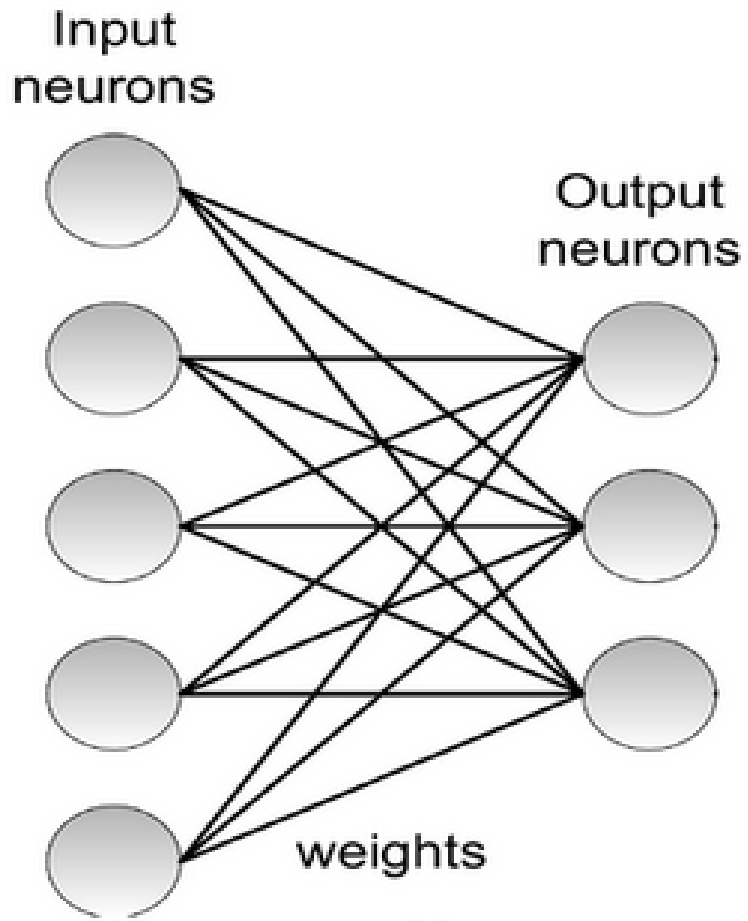
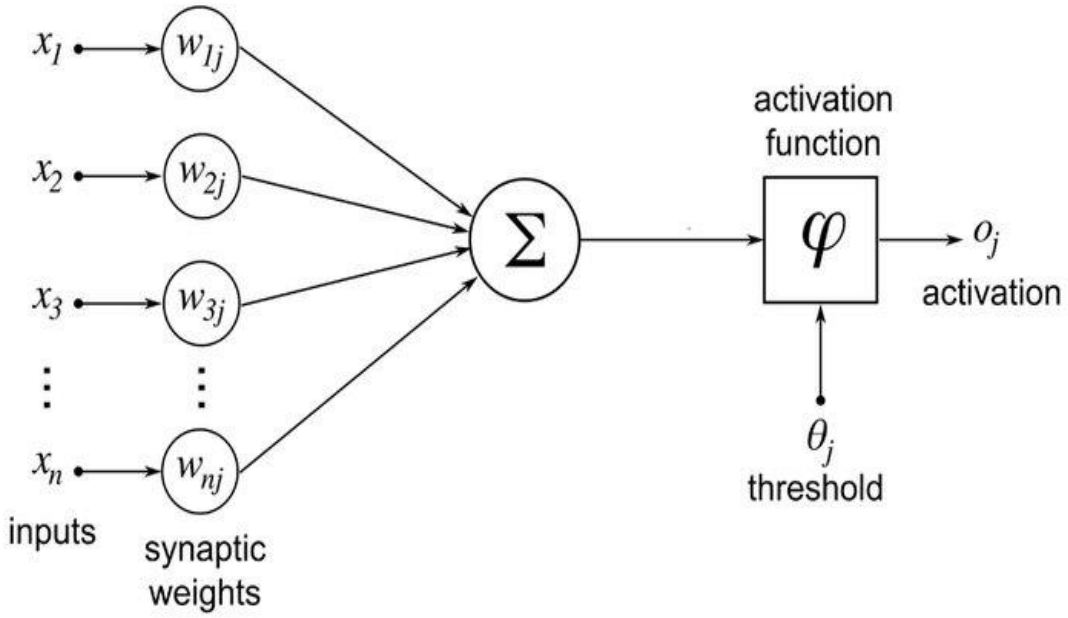
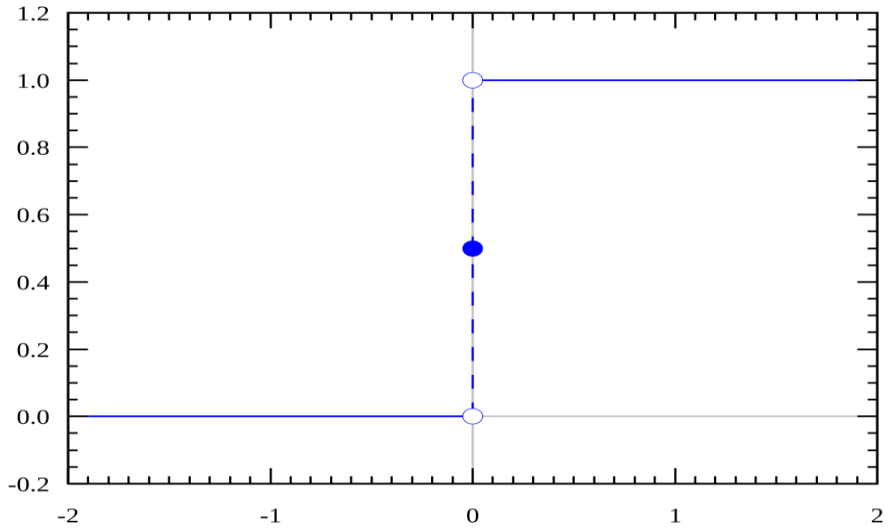


Fig source: Neuromorphic Spiking Neural Networks and Their Memristor-CMOS Hardware Implementations

An artificial neuron





Activation function: Heaveside step function

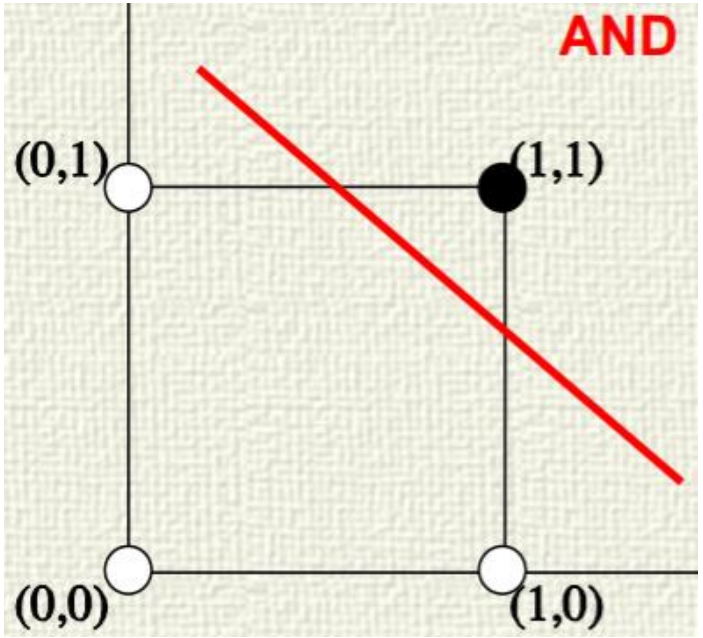
$\Omega(x) = 1$ if $x > 0$

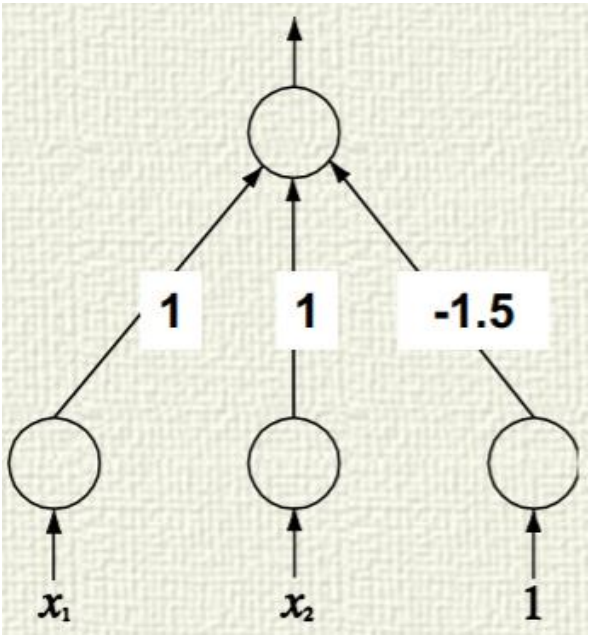
$\Omega(x) = 0$ if $x <= 0$

Fig source: Neuromorphic Spiking Neural Networks and Their Memristor-CMOS Hardware Implementations
https://en.wikipedia.org/wiki/Step_function

Representational Power of Perceptrons

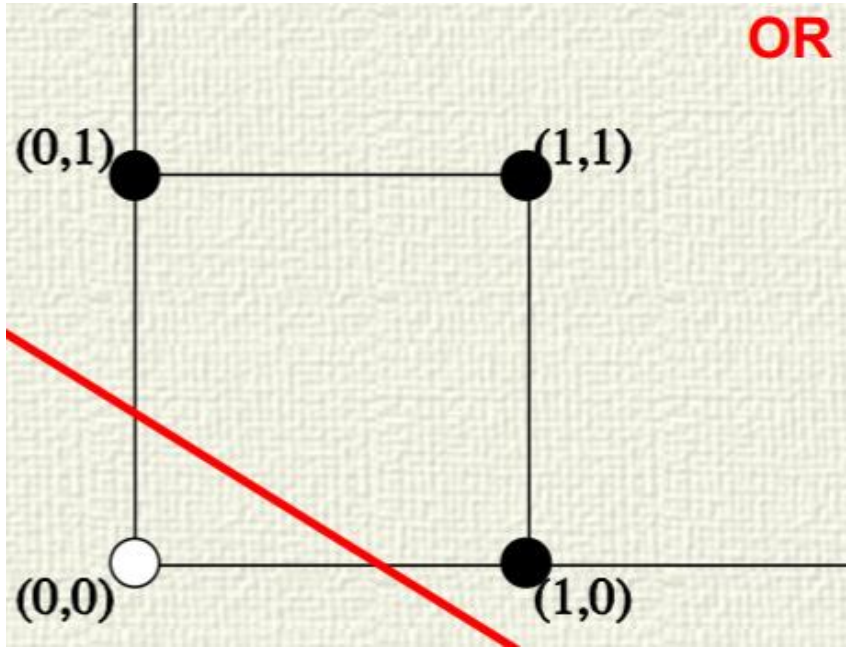
- Perceptron can solve linearly separable boolean functions

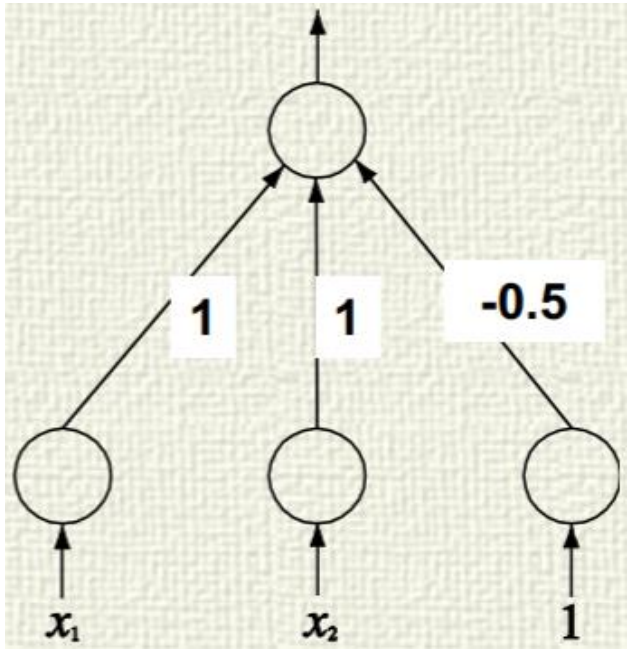




Representational Power of Perceptrons

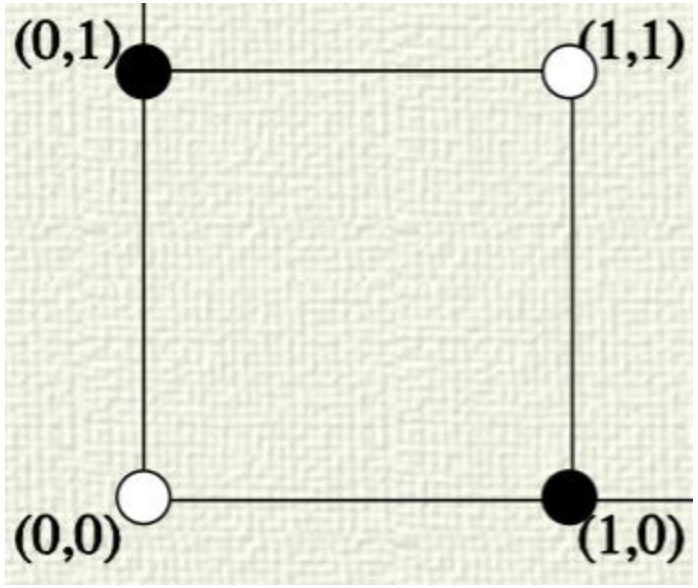
- Perceptron can solve linearly separable boolean functions





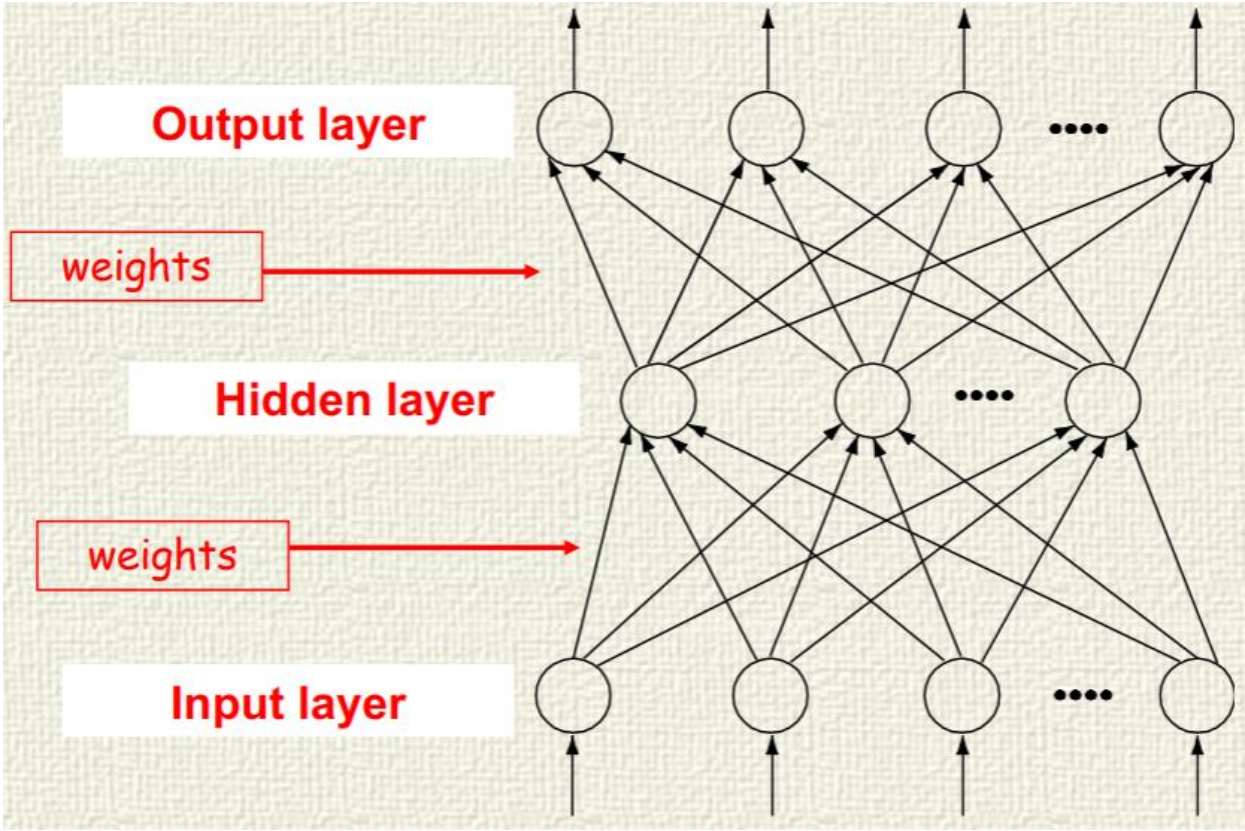
Representational Power of Perceptrons

- Perceptron cannot solve nonlinearly separable boolean functions



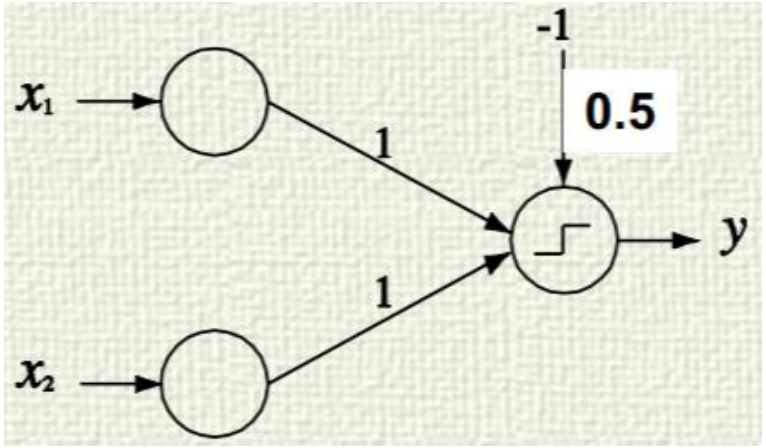
Solution: Add a hidden layer

- Multilayer Neural Network
  - Generalized perceptron having more computational power
  - Allows to learn nonlinearly separable transformations from input to output
  - A single hidden layer allows to compute any input/output transformation



Example: XOR

- First consider a perceptron that cannot solve xor



$$x_1 = 0, x_2 = 0 \quad H(-0.5) = 0$$
$$x_1 = 1, x_2 = 0 \quad H(1 - 0.5) = 1$$
$$x_1 = 0, x_2 = 1 \quad H(1 - 0.5) = 1$$

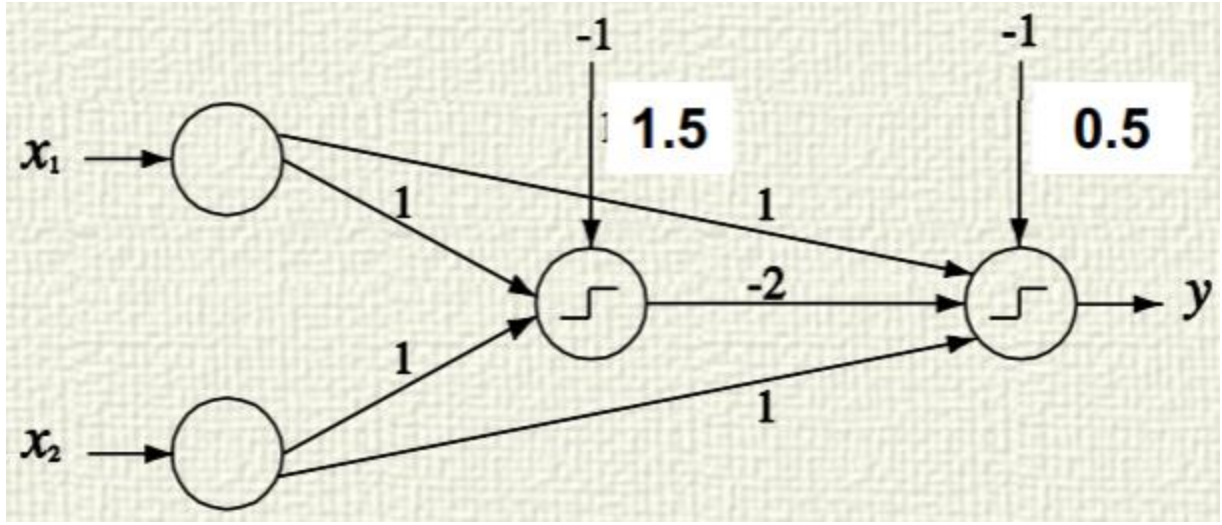
Right

$$x_1 = 1, x_2 = 1 \quad H(1 + 1 - 0.5) = 1$$

Wrong

Example: XOR

- Idea: Introduce one hidden unit with a large enough threshold, so that it is activated only in the 4th case.
- The hidden unit provides a negative input to the output unit to correct its response in the 4th case



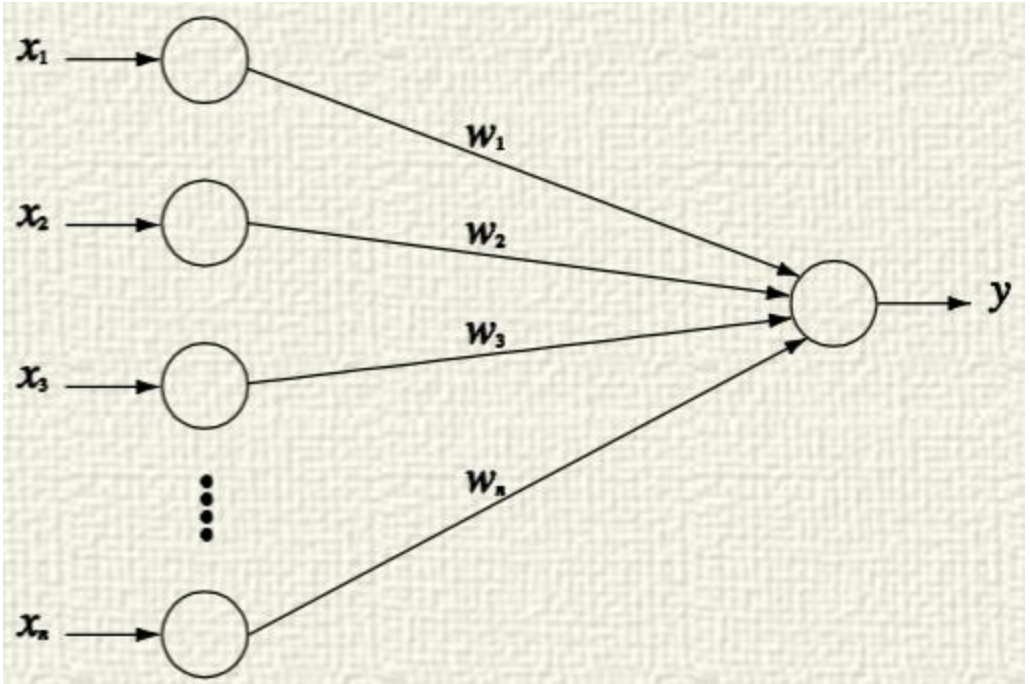
$$x_1 = 1, x_2 = 1 \quad H(1+1-2-0.5) = 0 \quad \textbf{OK!}$$

Neural Network Learning Framework

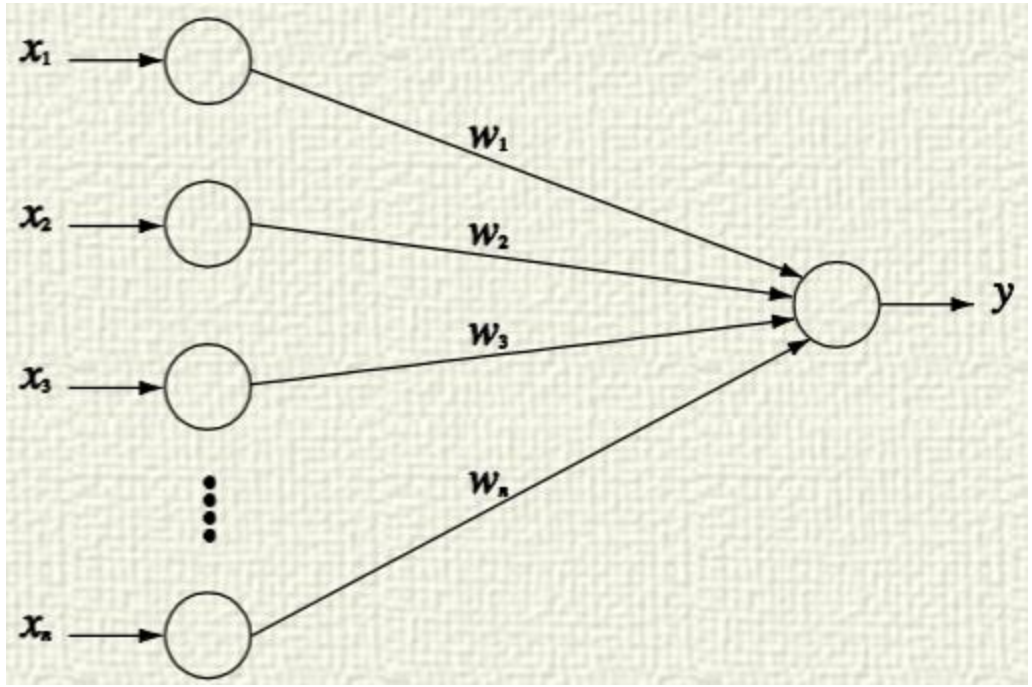
- Activation function

$$y = g\left(\sum_k w_k x_k\right) \in (0,1)$$

  - g must be a differential function
- NN dynamics:
  - The network will learn the transformation (f which is unknown) from input to output
  - The network adjusts the weights so that, after a finite number of steps, the outputs y ~ f(x)
- Criterion: Minimize the different between network's output and desired output



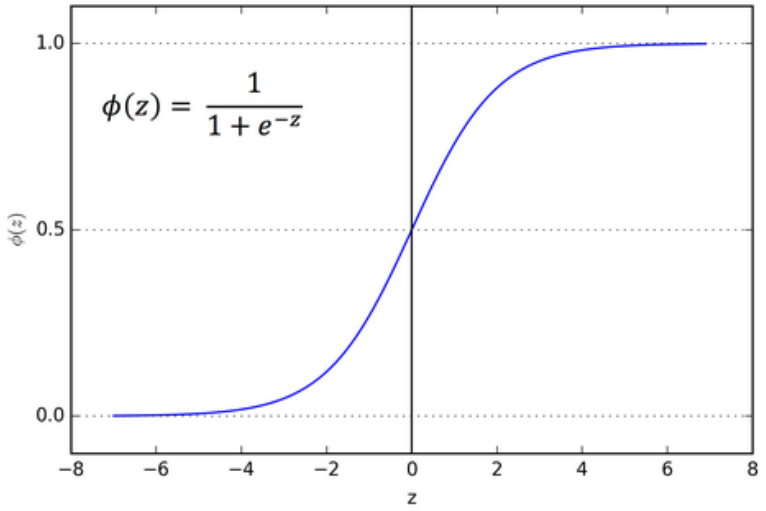
Learning Algotrithm: No hidden units



$$z = \boldsymbol{w}^T \boldsymbol{x}$$

$$y = g(z)$$

$$g(z) = \frac{1}{1+e^{-z}}$$



Sigmoid activation function

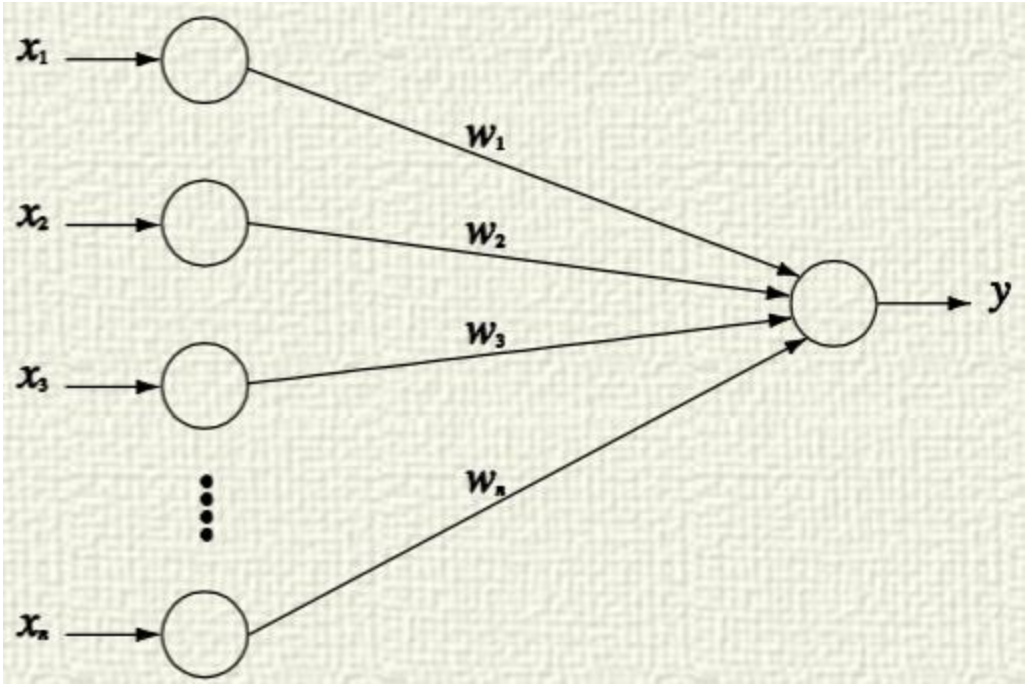
Learning Algotrithm: No hidden units

$$J(\boldsymbol{w}) = \frac{1}{2}(y^* - y)^2$$

$$y = g(\boldsymbol{w}) = \frac{1}{1 + e^{-\boldsymbol{w}^T \boldsymbol{x}}}$$

$$\frac{\partial y}{\partial w_i} = y(1-y)x_i$$

$$\frac{\partial J}{\partial w_i} = -(y^* - y)y(1-y)x_i$$



Weight update rule

- Delta rule
  - For the sigmoid activation function with no hidden layers

$$\Delta w_i = w_i^{t+1} - w_i^t$$

$$\Delta w_i = \mu y(1-y)(y^* - y)x_i$$

μ ***is the learning rate***

Acknowledgement

- Some figures are adopted from the Machine Learning class by Dr. Carlotta Dominiconi, George Mason University.