CS60010: Deep Learning
Spring 2021

Sudeshna Sarkar and Abir Das

Module 2

Part 3

Multilayer Perceptron - Introduction

Sudeshna Sarkar

18 Jan 2021

Biological Neural Network

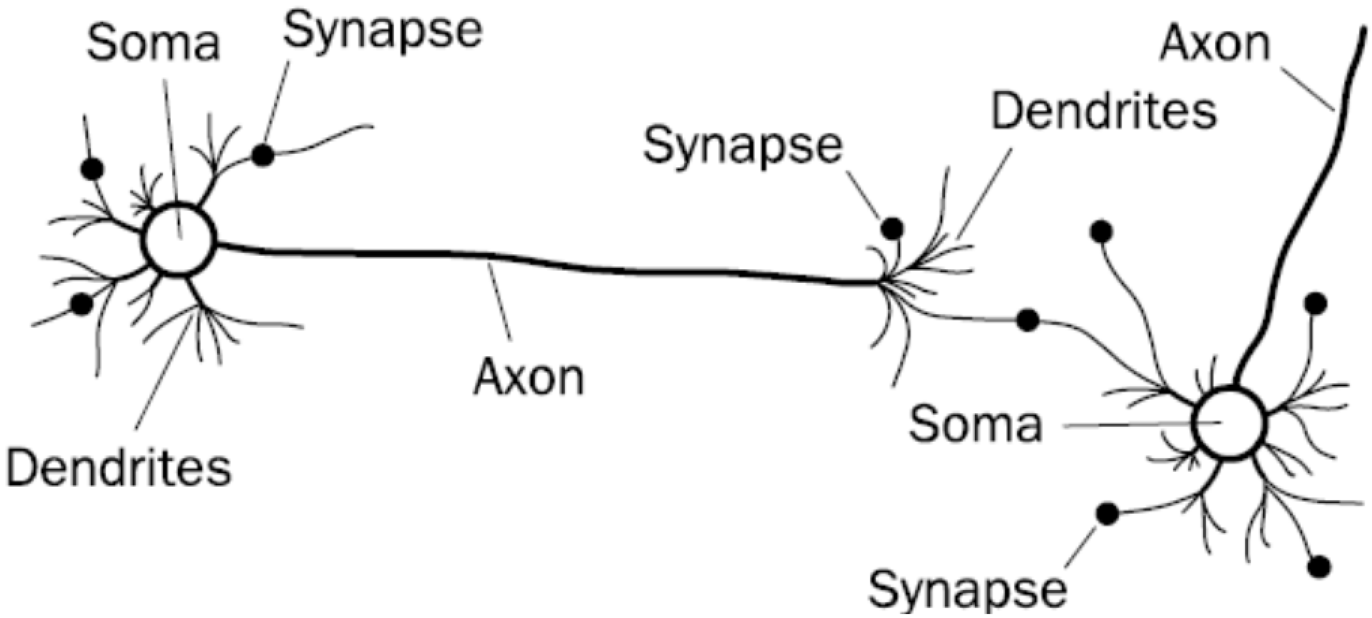


Image courtesy: F. A. Makinde et. al., “Prediction of crude oil viscosity using feed-forward back-propagation neural network (FFBPNN)”." Petroleum & Coal 2012

Phases of Neural Network Research

- 1940s-1960s: Cybernetics: Brain like electronic systems, morphed into modern control theory and signal processing.
  - McCulloch & Pitts; Hebb: modeling real neurons

Phases of Neural Network Research

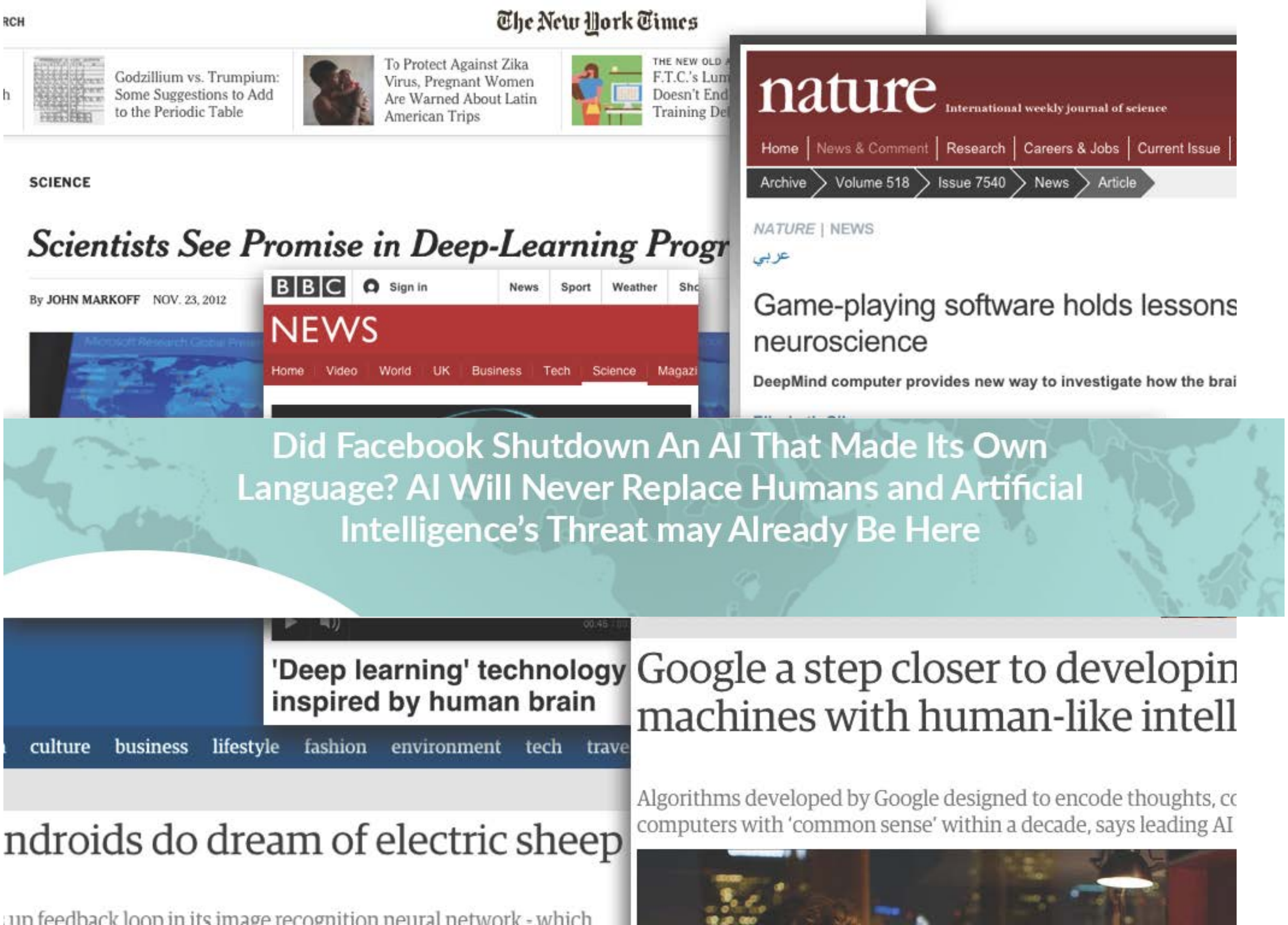
- 1940s-1960s: Cybernetics: Brain like electronic systems, morphed into modern control theory and signal processing.
  - McCulloch & Pitts; Hebb: modeling real neurons
  - Rosenblatt, Widrow-Hoff: : perceptrons
  - 1969: Minskey & Papert, *Perceptrons* book showed formal limitations of one-layer linear network

Phases of Neural Network Research

- 1960s-1980s: **D**igital computers, automata theory, computational complexity theory: simple shallow circuits are very limited…
- 1980s-1990s: Connectionism: complex, non-linear networks, back-propagation.
  - backprop and multi-layer networks
  - Rumelhart and McClelland *PDP* book set
  - Sejnowski’s NETTalk, BP-based text-to-speech
  - Neural Info Processing Systems (NIPS) conference starts
- 1990s-2010s: Computational learning theory, graphical models: Learning is computationally hard, simple shallow circuits are very limited…

Phases of Neural Network Research

- 2006: Deep learning: End-to-end training, large datasets, explosion in applications.



Did Facebook Shutdown An AI That Made Its Own Language? AI Will Never Replace Humans and Artificial Intelligence's Threat may Already Be Here

- Layered architecture (the deep part) of simple units.
- Inner layer representations are learned only from end-to-end tasks.
- Depth and complexity seem to be only limited by the amount of data. More complex models → better representations → better accuracy.
- This behavior is fundamentally different from classical ML: there is often no obvious performance ceiling.
- Inner layer representations are typically task-independent → easy to re-use models for applications that don't have large training datasets.
- Multi-task learning usually works: another departure from typical behavior of classical ML methods.

The success of NN

Yann Lecun: DNNs require: “an interplay between intuitive insights, theoretical modeling, practical implementations, empirical studies, and scientific analyses”

1. More data
2. More computational power
3. Improved techniques (though they’re not brand-new)

   But, Driven primarily by intuition and empirical success

- Good research and progress based on
  - Intuition, Practice (empirical findings)
- Theory lags dramatically
  - No guarantees, little understanding of limitations, limited interpretability
- More interestingly, classic theory suggests currently successful DL practices, wouldn’t be likely to succeed.

Why the success of DNNs is surprising

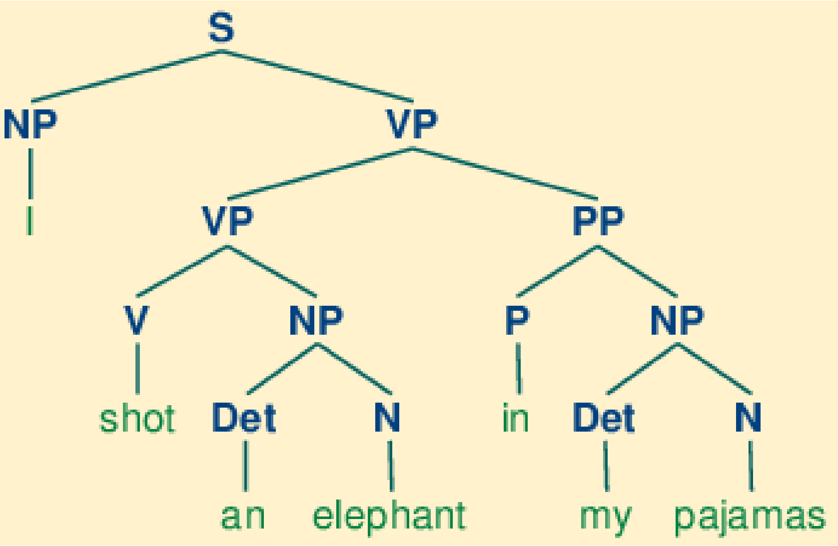
- The most successful DNN training algorithm is a version of gradient descent which will only find local optima. In other words, it's a greedy algorithm. Backprop:

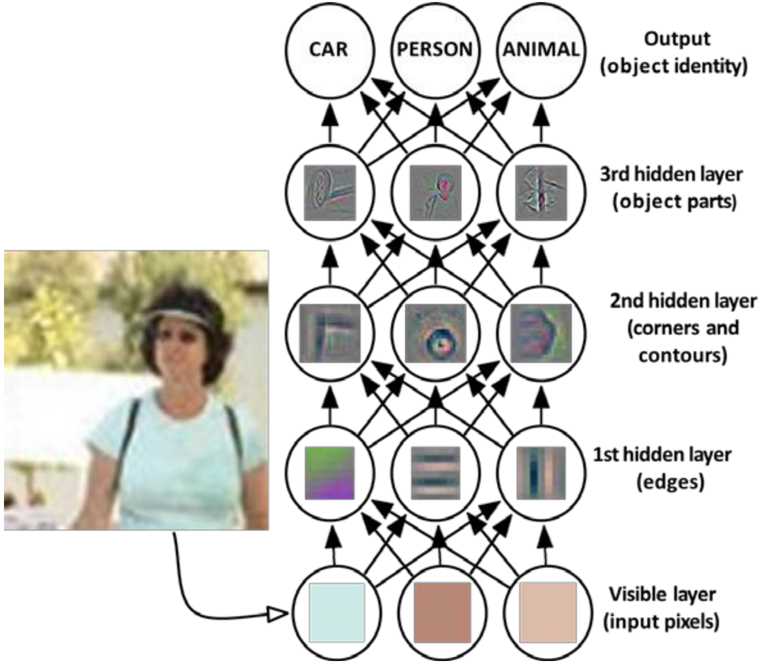
$$\text{Loss} = f\left(g\big(h(y)\big)\right)$$

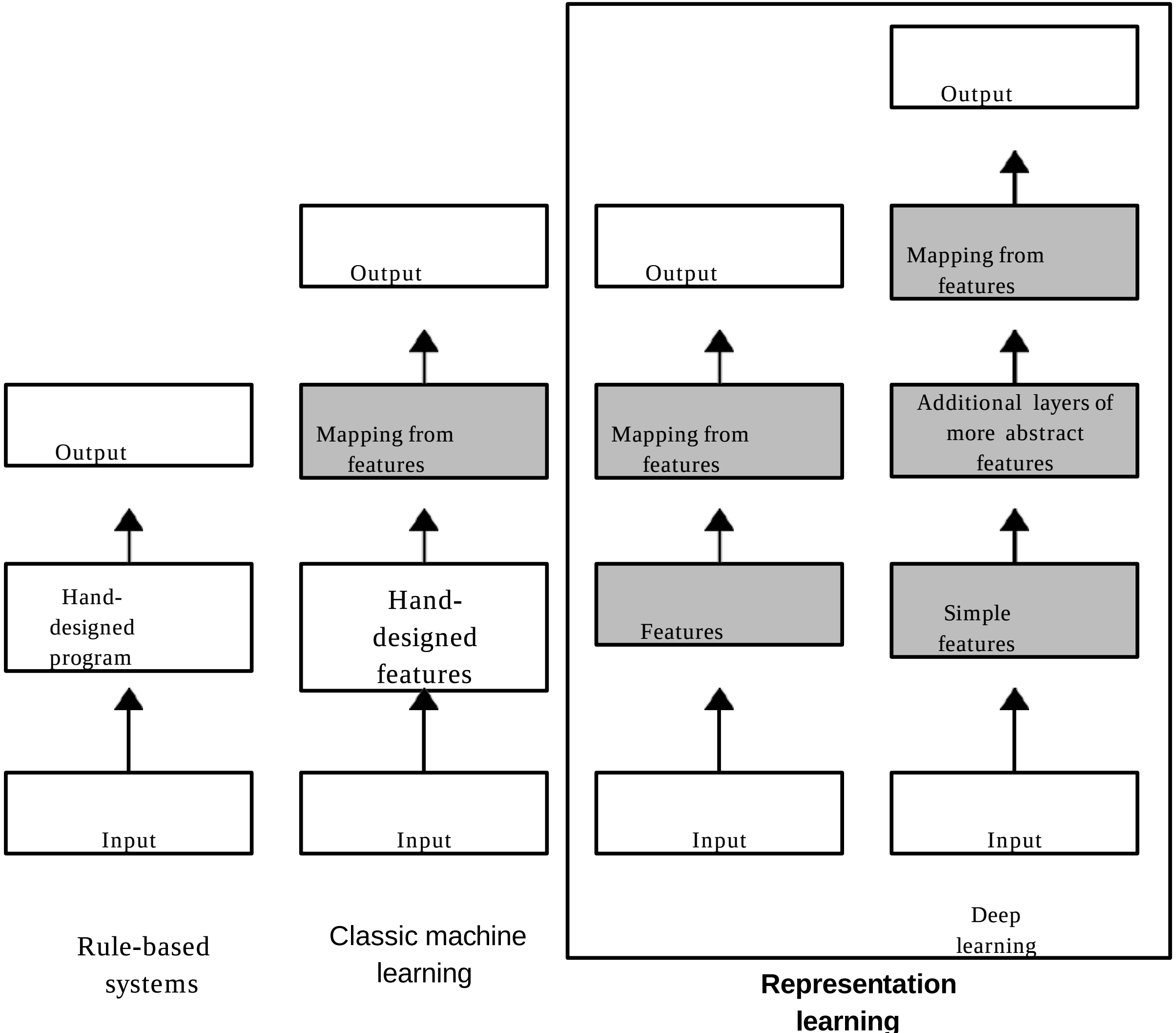
$$\frac{d\,\text{Loss}}{dy} = f'(g) \times g'(h) \times h'(y)$$

Why the success of DNNs

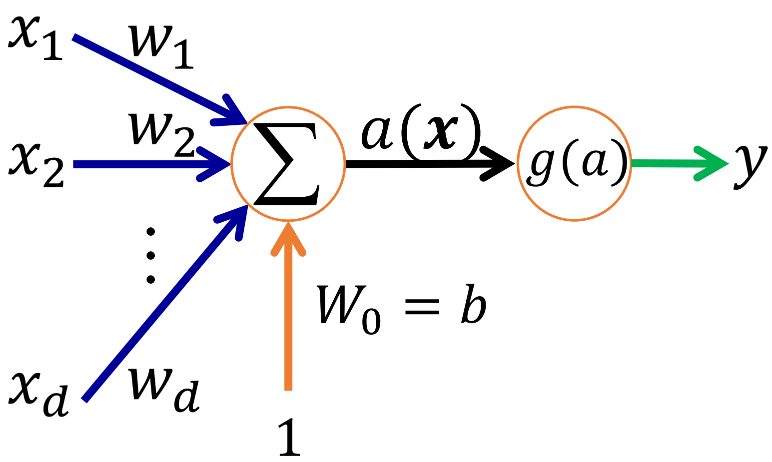
Hierarchical representations are ubiquitous.






Output
Hand-designed program
Input
Rule-based systems
Output
Mapping from features
Hand-designed features
Input
Classic machine learning
Output
Mapping from features
Features
Input
Output
Mapping from features
Additional layers of more abstract features
Simple features
Input
Deep learning
Representation learning

Artificial Neuron



$$\boldsymbol{w} = [w_1\ w_2 \dots w_d]^T \text{ and } \boldsymbol{x} = [x_1\ x_2 \dots x_d]^T$$

$$a(\boldsymbol{x}) = b + \sum_{i=1}^{d} w_i x_i = [\boldsymbol{w}^T b] \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix}$$

$$y = g(a(\boldsymbol{x}))$$

<u>Terminologies</u>:-

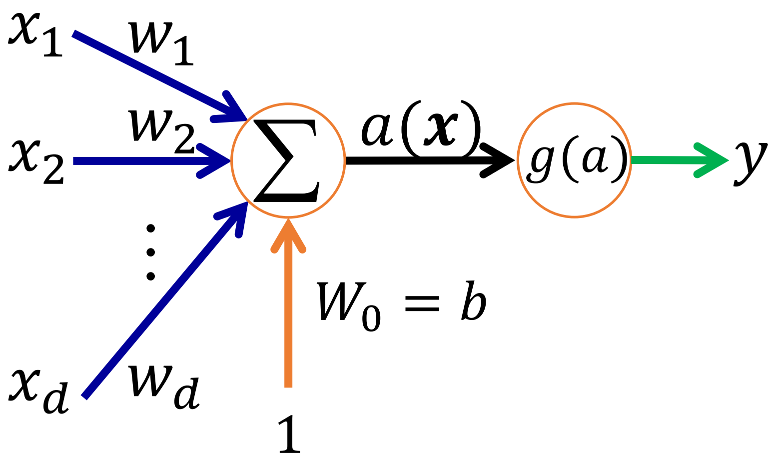
$\boldsymbol{x}$**:** input, $\boldsymbol{w}$**:** weights, $\boldsymbol{b}$**:** bias

a: pre-activation (input activation)

g: activation function

y: activation (output activation)

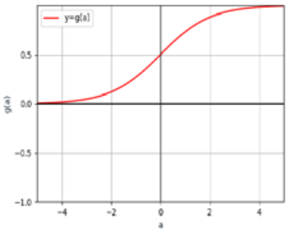
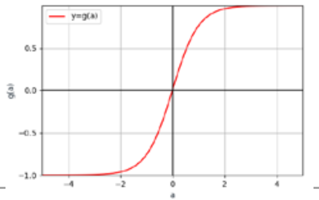
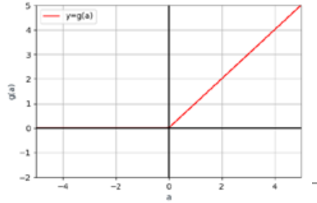
Perceptron



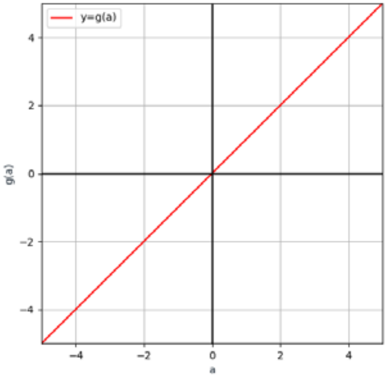
$\boldsymbol{x} \in \mathcal{R}^d$ and $y \in \{0, 1\}$ for Binary Classification

$$g(a) = \begin{cases} 1, & a \geq 0 \\ 0, & a < 0 \end{cases}$$ (Rosenblatt, 1957)

Common Activation Functions

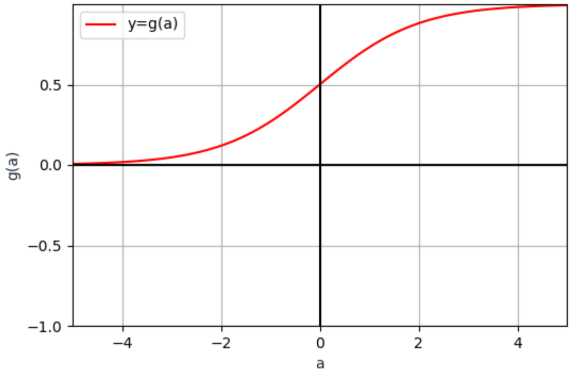
Name	Function	Gradient	Graph
Binary step	$sign(a)$	$g'(a) = \begin{cases} 0, & a \neq 0 \\ NA, & a = 0 \end{cases}$	
Sigmoid	$\sigma(a) = \dfrac{1}{1+\exp(-a)}$	$g'(a) = g(a)(1-g(a))$	
Tanh	$\tanh(a) = \dfrac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)}$	$g'^{(a)} = 1 - g^2(a)$	
ReLU	$g(a) = \max(0, a)$	$g'(a) = \begin{cases} 1, & a \geq 0 \\ 0, & a < 0 \end{cases}$	

Common Activation Functions



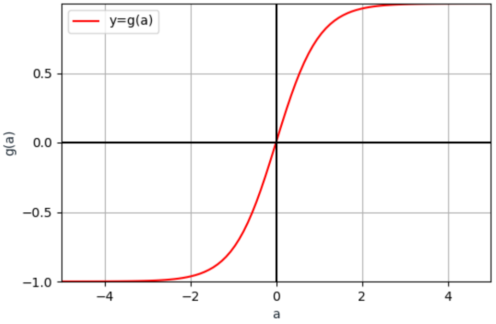
Linear activation function

- $g(a) = a$
- Unbounded
- $g'(a) = 1$



Sigmoid activation function

- $g(a) = \sigma(a) = \frac{1}{1+\exp(-a)}$
- Bounded (0, 1)
- Always positive
- $g'(a) = g(a)(1 - g(a))$

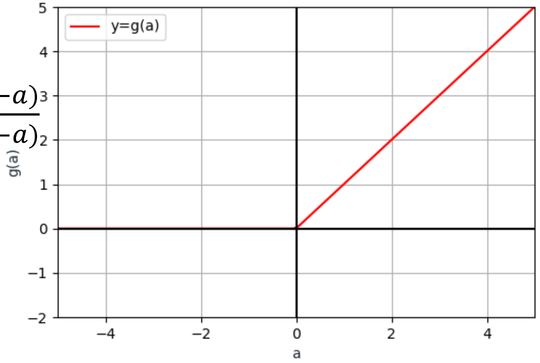


tanh activation function

- $g(a) = tanh(a)$

$$= \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)}$$

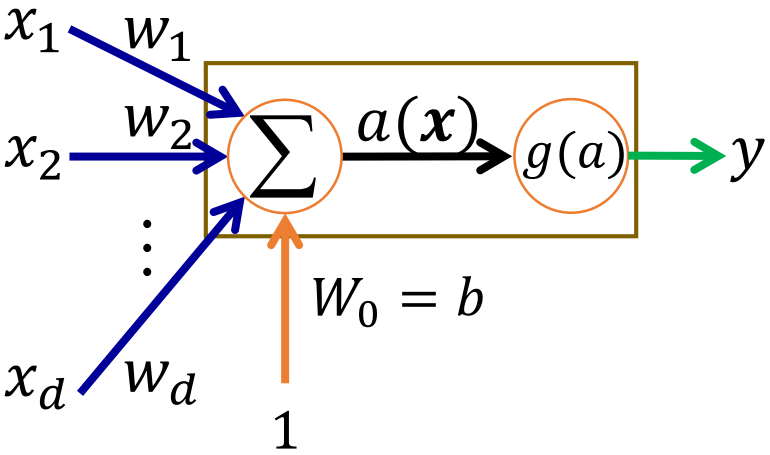
- Bounded (-1, 1)
- Can be positive or negative
- $g'(a) = 1 - g^2(a)$

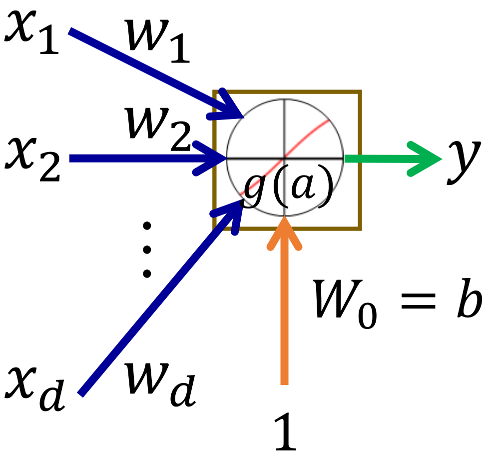


ReLU activation function

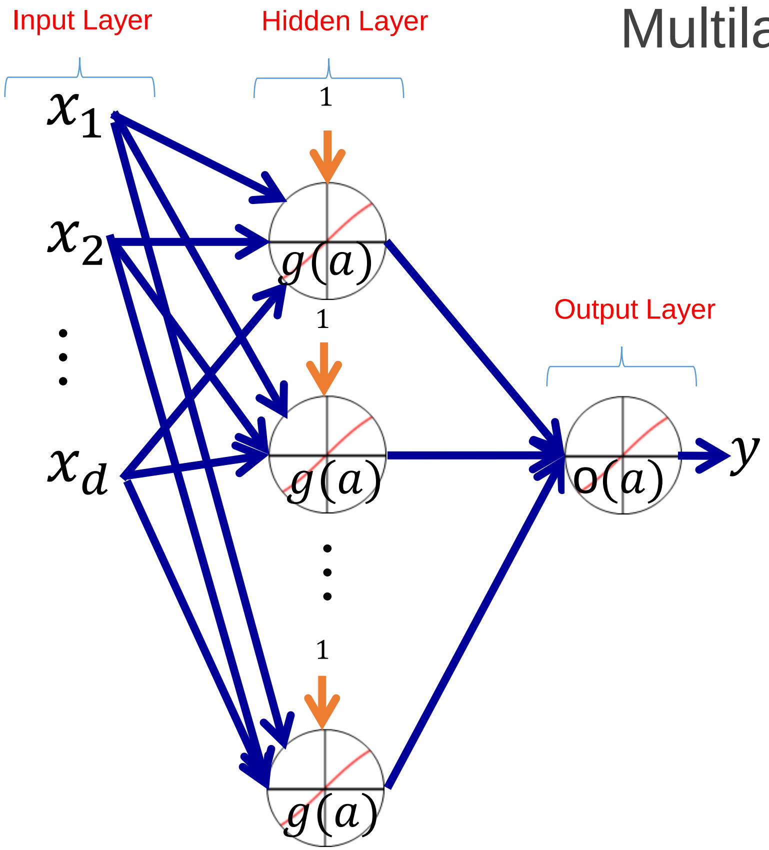
- $g(a) = \max(0, a)$
- Bounded below by 0
- But not upper-bounded
- $g'(a) = \begin{cases} 1, & a \geq 0 \\ 0, & a < 0 \end{cases}$

Multilayer Neural Network





Multilayer Neural Network



Intuition:-
Hidden Layer: Extracts better representation of the input data
Output layer: Does the classification