Class
Artificial Intelligence

Ari Barrera, Ph.D.

May 2023

UNIVERSIDAD Panamericana

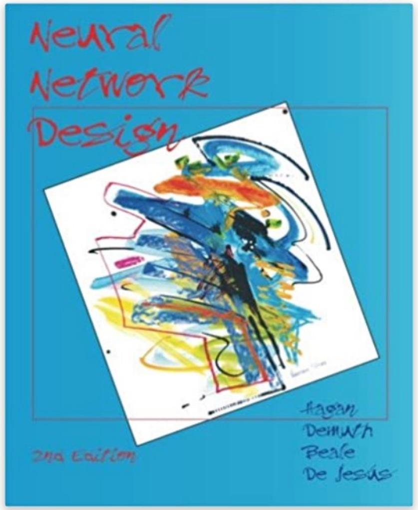
Machine Learning

Supervised learning

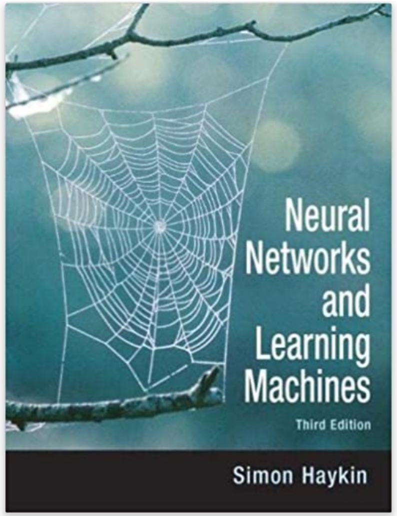
Neural Networks

Neural Networks

- Bibliography
  - Hagan, M. T., Demuth, H. B., & Beale, M. (1997). Neural network design. PWS Publishing Co.
  - Haykin, S. S. (2009). Neural networks and learning machines/Simon Haykin.



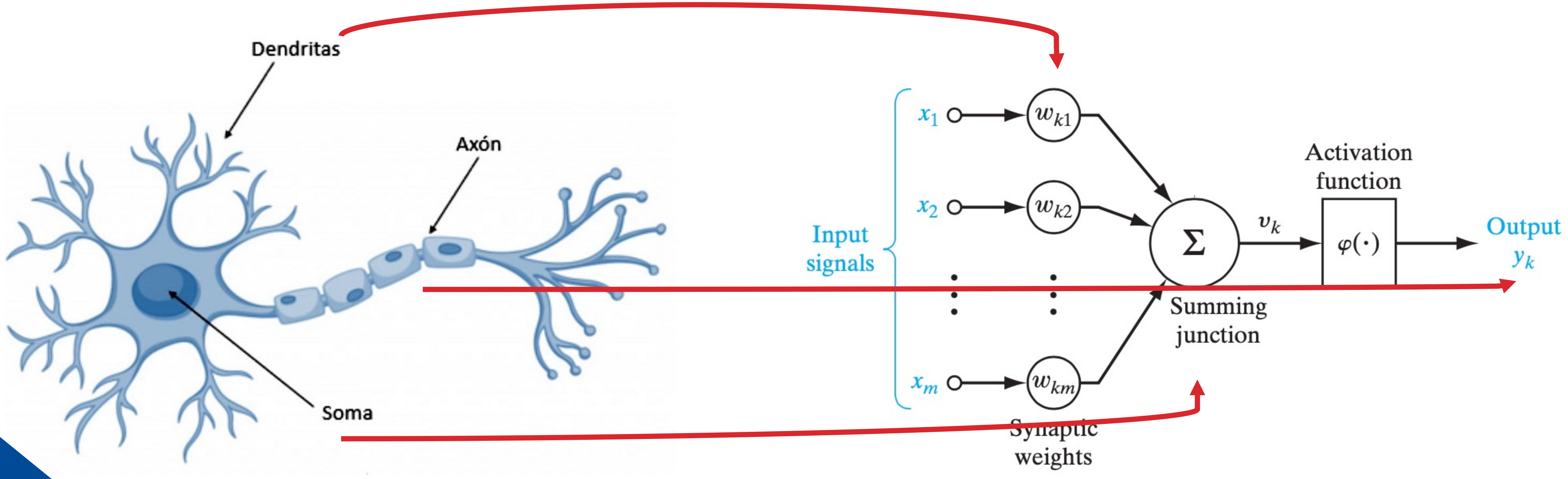
Taken from https://www.amazon.com/Neural-Network-Design-Martin-Hagan/dp/0971732116



Taken from https://www.amazon.com/Neural-Networks-Learning-Machines-3rd/dp/0131471392

Neural Networks

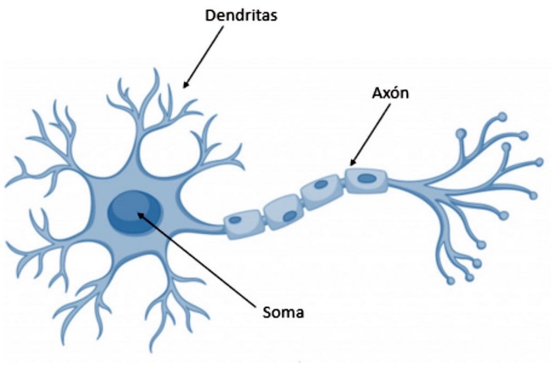
- From Biological Neuron to Artificial Neuron



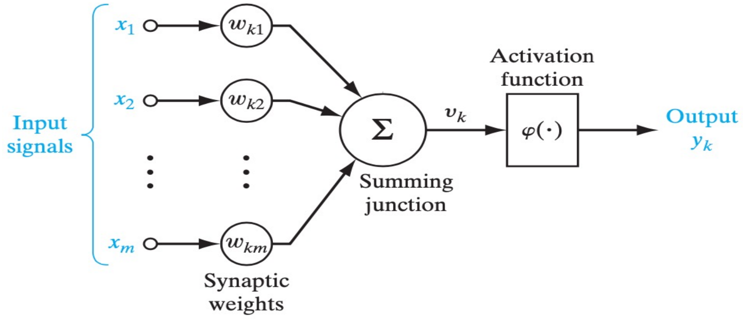
Taken from https://www.xeridia.com/blog/redes-neuronales-artificiales-que-son-y-como-se-entrenan-parte-i

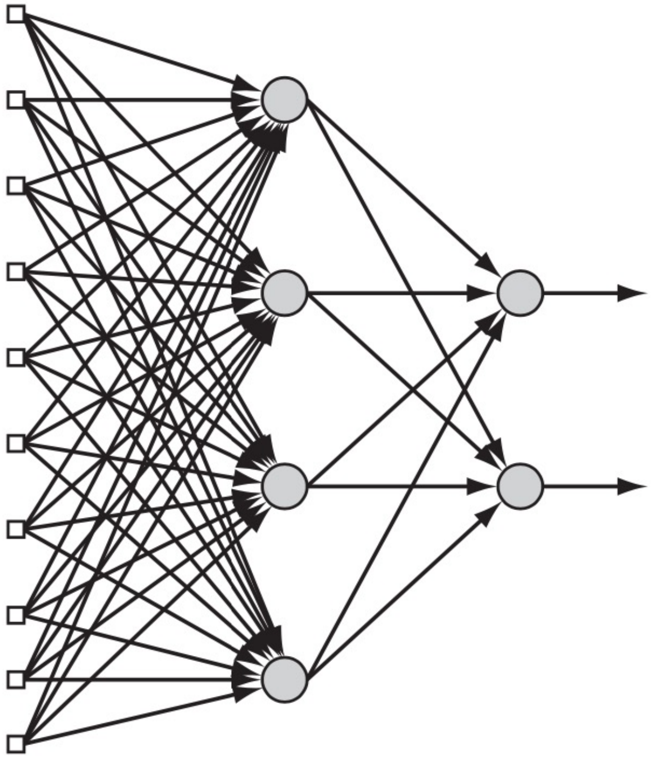
Neural Networks

- From Biology to Artificial Neural Networks



Taken from https://www.xeridia.com/blog/redes-neuronales-artificiales-que-son-y-como-se-entrenan-parte-i





Neural Networks

Basic Concepts

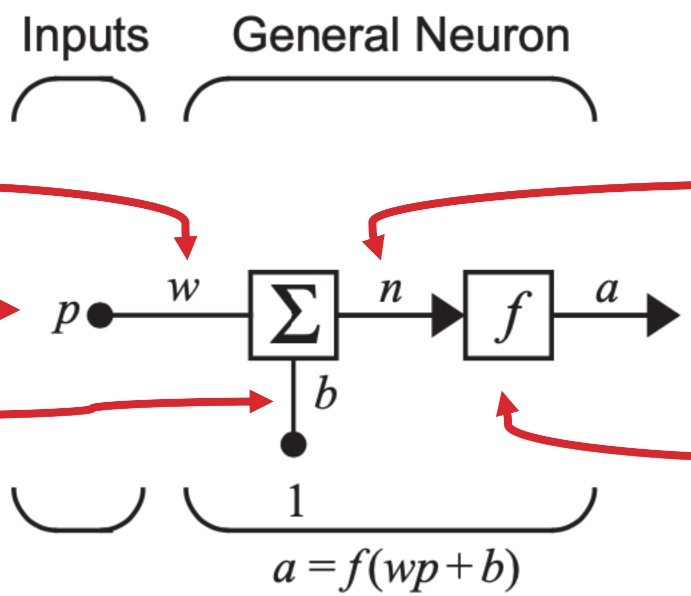
Neural Networks

- **Basic Concepts – Single Input Neuron**

Weight (adjustable scalar)

Scalar input

Bias/offset (adjustable scalar)



Summer output or net input

Scalar neuron output

Transfer/Activation function (to be chosen)

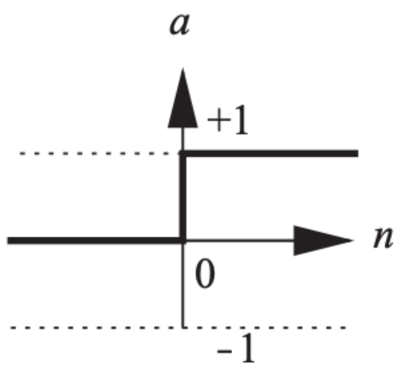
Example: if $w = 3, p = 2, and\ b = -1.5$

$$a = f(3 * 2 \ - 1.5)$$
$$a = f(4.5)$$

Neural Networks

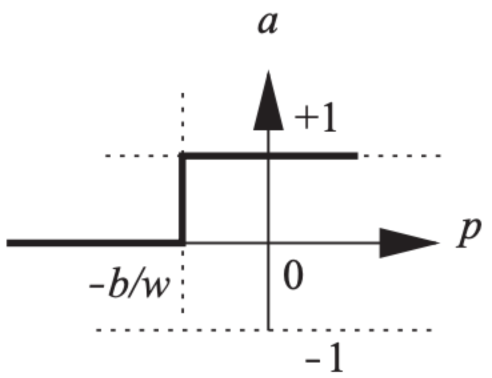
- **Basic Concepts – Transfer/Activation functions**

- A particular transfer function is chosen to satisfy some specification of the problem that the neuron is attempting to solve



$a = hardlim(n)$

Hard Limit Transfer Function



$a = hardlim(wp+b)$

Single-Input *hardlim* Neuron

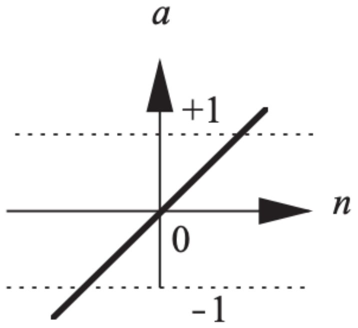
$$a = f(n) = \begin{cases} 0, & n < 0 \\ 1, & n \geq 0 \end{cases}$$

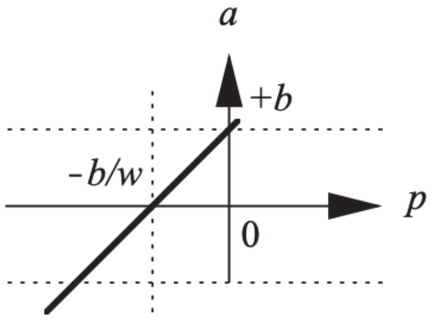
This function to create neurons that classify inputs into two distinct categories

Neural Networks

UNIVERSIDAD Panamericana

- **Basic Concepts – Transfer/Activation functions**





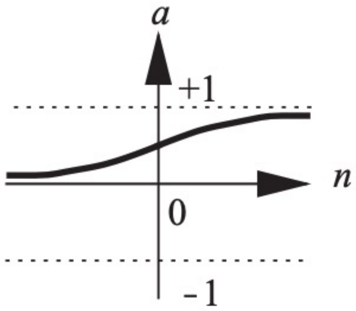
$a = purelin(n)$

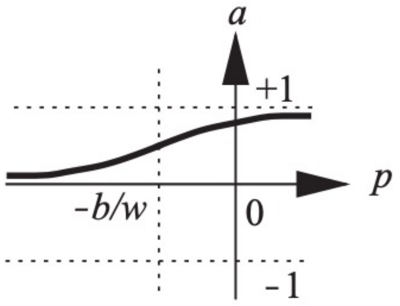
Linear Transfer Function

$a = purelin(wp + b)$

Single-Input *purelin* Neuron

This transfer function takes the input $_{-\infty}^{\infty}n$ and squashes the output to $_0^1 a$ according to $a = \frac{1}{1+e^{-n}}$





$a = logsig(n)$

Log-Sigmoid Transfer Function

$a = logsig(wp + b)$

Single-Input *logsig* Neuron

The log-sigmoid transfer function is commonly used in multilayer networks that are trained using the backpropagation algorithm

Neural Networks

- **Basic Concepts – Transfer/Activation functions**

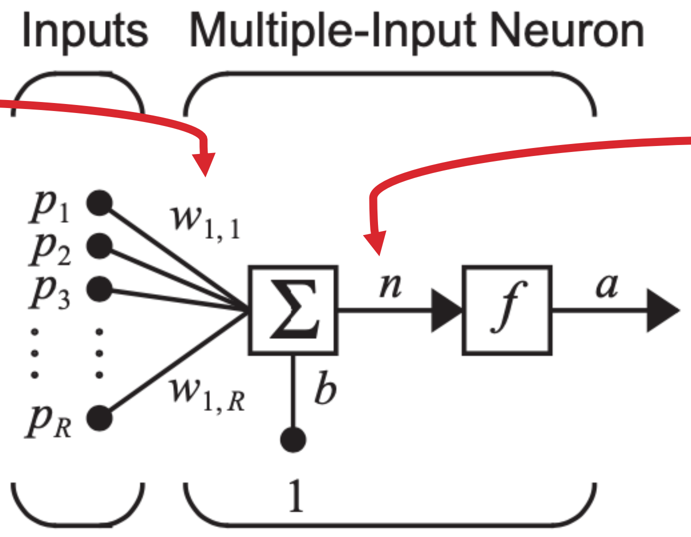
Name	Input/Output Relation	Icon
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$	
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$	
Linear	$a = n$	
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$	
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$	

Name	Input/Output Relation	Icon
Log-Sigmoid	$a = \frac{1}{1 + e^{-n}}$	
Hyperbolic Tangent Sigmoid	$a = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}}$	
Positive Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n$	
Competitive	$a = 1$ neuron with max n $a = 0$ all other neurons	C

Neural Networks

- **Basic Concepts – Multiple Input Neuron**

Weighted matrix
$w_{1,1}, w_{1,2}, \dots, w_{1,R}$



$a = f(\mathbf{Wp} + b)$

$$n = w_{1,1} * P_1 + w_{1,2} * P_2 + \dots + w_{1,R} * P_R + b$$

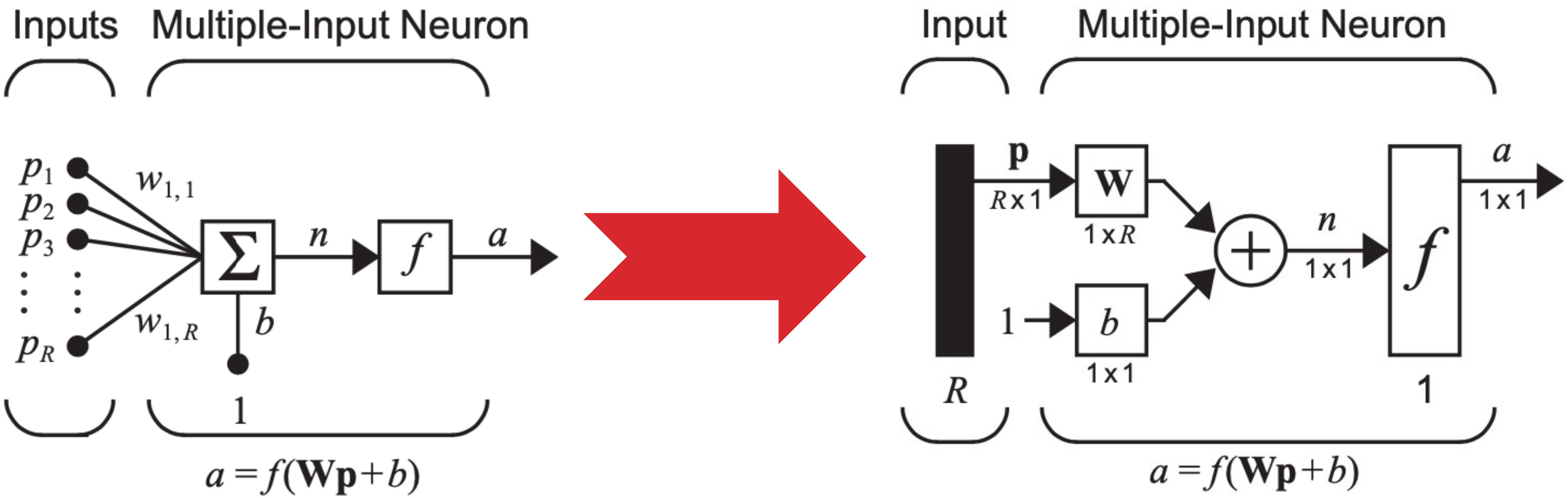
or

$$n = Wp$$

Fortunately, neural networks can often be described with matrices

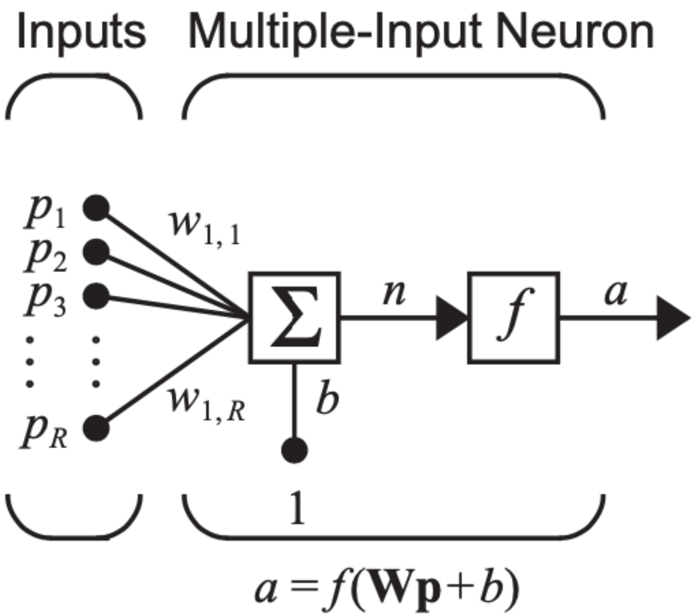
Neural Networks

- **Basic Concepts – Multiple Input Neuron**



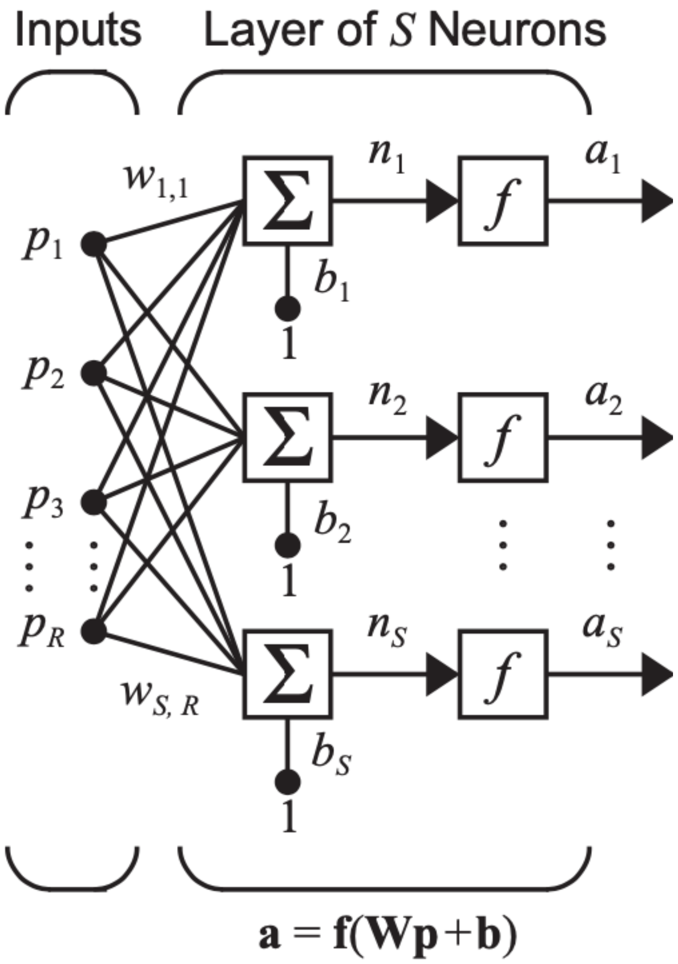
Neural Networks

- **Basic Concepts – Layer of Neurons**



A single-*layer* network of 1 neuron

A single-*layer* network of *S* neurons



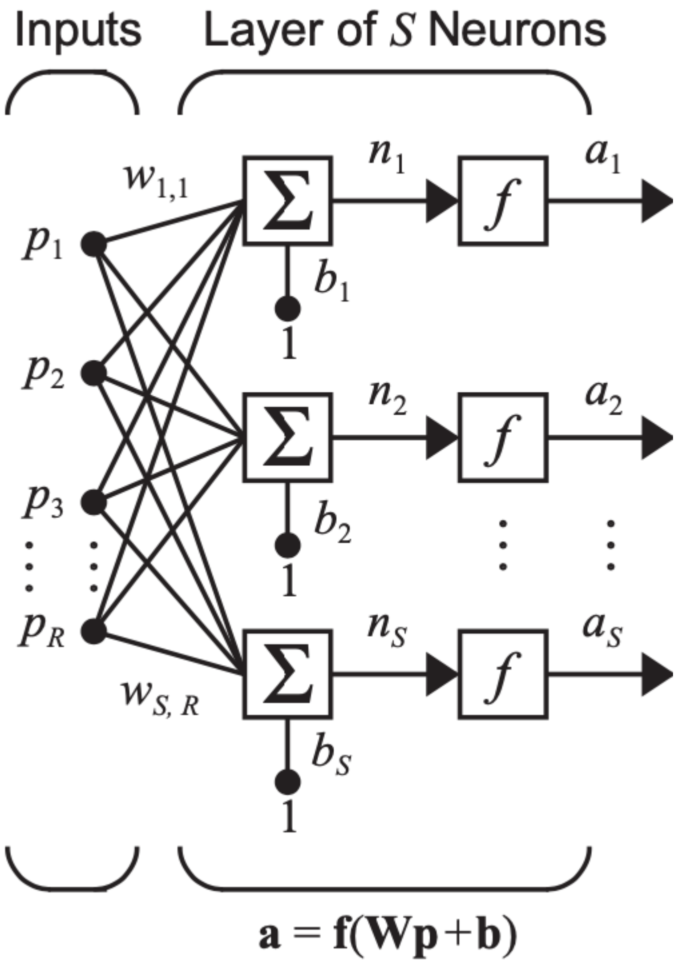
Neural Networks

- **Basic Concepts – Layer of Neurons**

- It is common for the number of inputs to a layer to be different from the number of neurons (i.e., $R \neq S$)

- The transfer/activation function can be different for all neurons in a layer

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

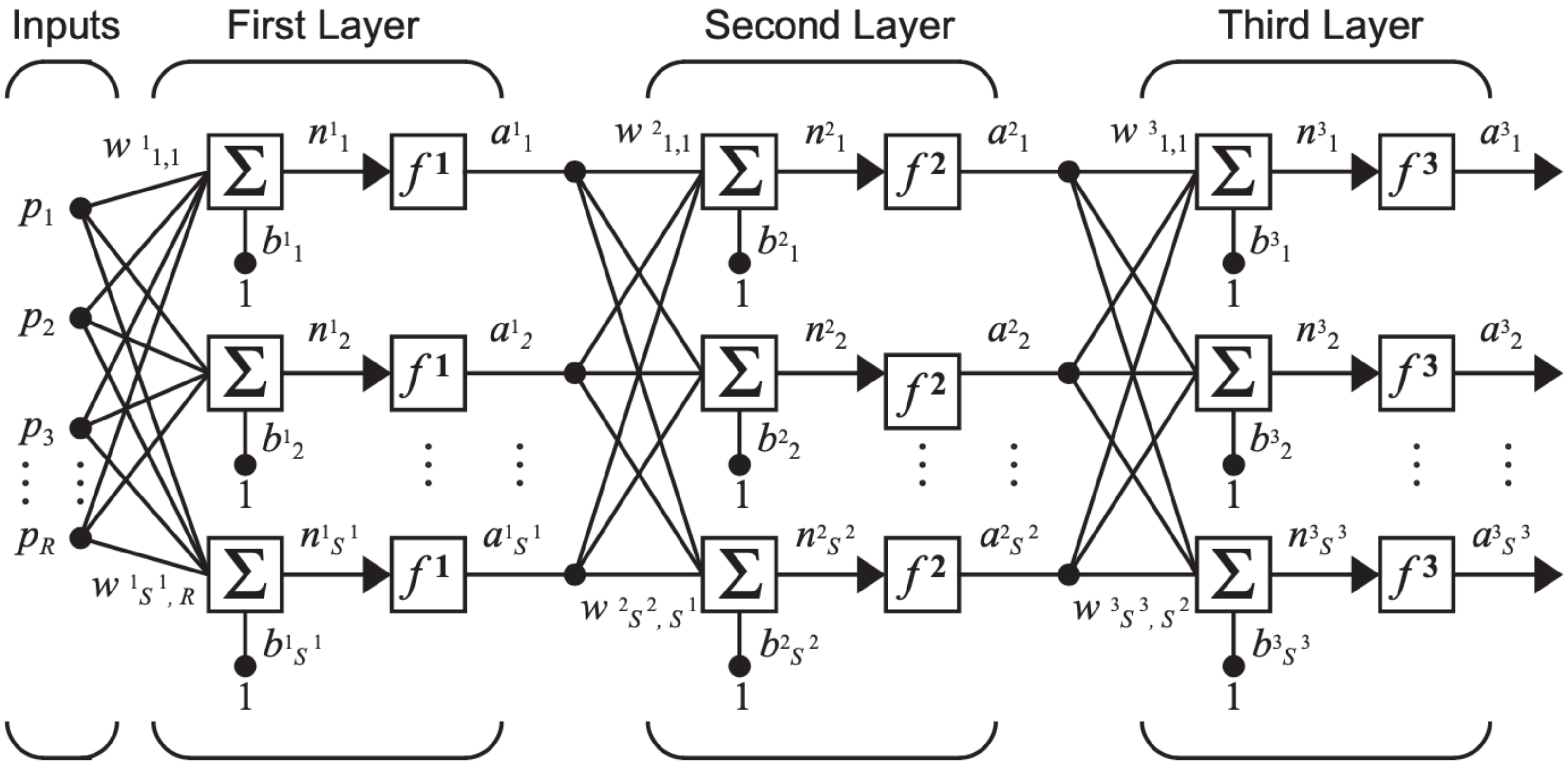


Neural Networks

- **Basic Concepts – Multiple layers of Neurons**

Each layer has its own :

- weight matrix **W**
- bias vector **b**
- a net input vector **n**
- an output vector **a**



$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)$ $\mathbf{a}^2 = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1+\mathbf{b}^2)$ $\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{a}^2+\mathbf{b}^3)$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

Neural Networks

- **Basic Concepts – Names of the layers and how to make choices**

There are few problems for which one can predict the optimal number of neurons needed in a hidden layer. This problem is an active area of research. Heuristically, in practice 2 and 3 layers NNs are used.

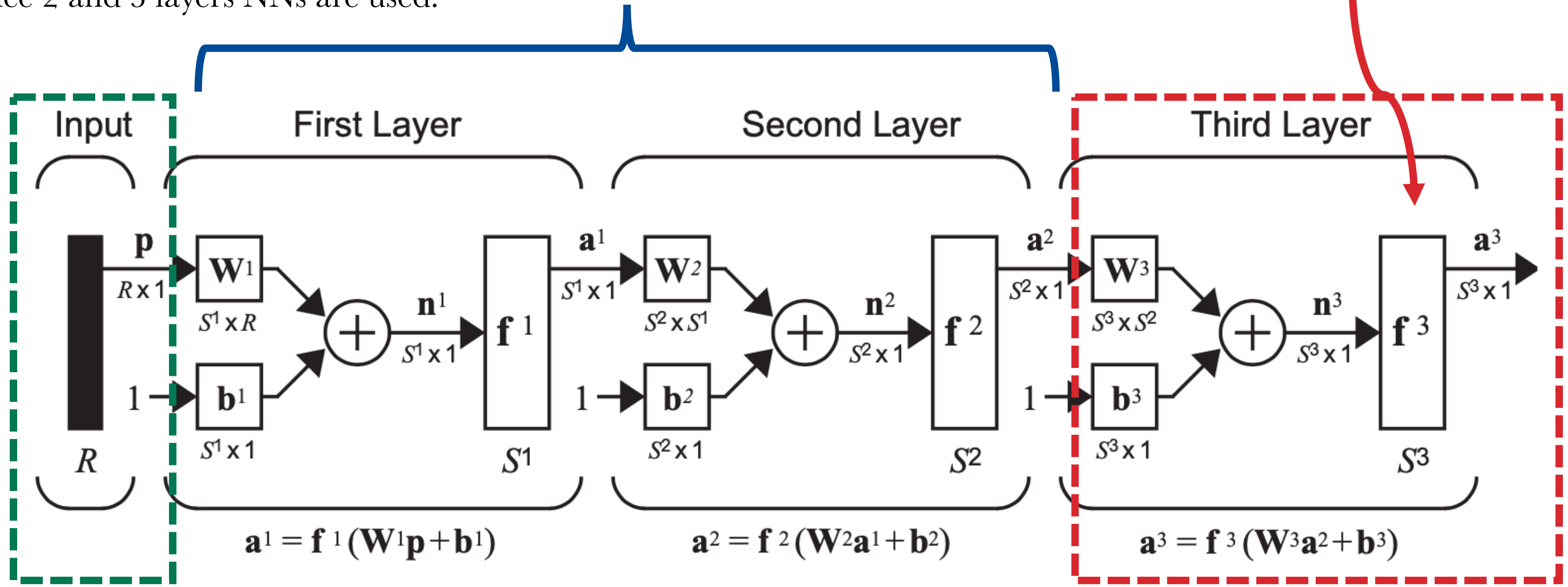
The desired characteristics of the output signal also help to select the transfer function for the output layer. If an output is to be either −1 or 1 , then a symmetrical hard limit transfer function should be used

Hidden layers

Input layer

Defined by external problem specifications

E.g., If there are four external variables to be used as inputs, there are four inputs to the network



$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

Output layer

Defined by external problem specifications

E.g., If there are to be seven outputs from the network, there must be seven neurons in the output layer.

Neural Networks

Basic Concepts

Types of Neural Networks

Neural Networks

- **Types of NNs**

- Single Layer Perceptrons
- Multilayer Perceptrons (MLPs)
- Radial-Basis Function Networks (RBFs)
- Hopfield Networks
- Boltzmann Machines
- Self-Organization Maps (SOMs)
- Modular Networks (Committee Machines)
- Support Vector Machines
- Bayesian Networks
- Probabilistic Graphical Models
- Hidden Markov Models

Neural Networks

Basic Concepts

Types of Neural Networks

Issues

Neural Networks

- **Issues**

- What is an appropriate architecture for a given learning problem?
  - Should units be divided into layers? How many?
  - What sort of activation function in units?
  - What type of updating? Incremental (stochastic), batch, synchronous, asynchronous?
  - How many units?
- How can the network be programmed?
  - What must be pre-designed? What can be learned?
  - How many samples are needed for good performance?
  - Is on-line learning possible?
  - What kind of learning information is available?

Neural Networks

- **Issues**

- What are the characteristics of a network?

  - For which problem is it suitable (what “function” can the network represent)?
  - Is it optimal in some sense?
  - What are the assumptions underlying the network? (structural biases, problem biases, noise distributions, etc.)
  - How robust is a network to violations of its assumptions?
  - Can the network generalize from a known task to unknown ones?
  - How hard is it to initialize the network?

Neural Networks

Basic Concepts

Types of Neural Networks

Issues

When to use them

Neural Networks

When to use them

- A large set of pairs are available as training examples
- Output values are discrete, continuous, or combinations of both
- Learning examples are noisy
- Long learning time is tolerable
- Fast execution is required
- Human interpretation of the learned model is not important

Neural Networks

Basic Concepts

Types of Neural Networks

Issues

When to use them

Practice Neural Networks

Neural Networks

- **Practice Neural Networks**

- We need the Neural_Networks_introduction.zip file

- It contains the following:

    - The Neural_Networks_introduction.ipynb notebook to be open with Google Colab

    - The Iris dataset data in the iris.data file

    - The Iris dataset names in the iris.names file