Al-Mustaqbal University

College of Engineering

Department of Biomedical Engineering

Stage: 5th Subject : Neural Eng.

2025-2026

Lecture ( ):

Artificial Neural Networks (ANN)

2025-2026

1. Introduction:

1. History:

The history of artificial neural networks is filled with colorful, creative individuals from a variety of fields, many of whom struggled for decades to develop concepts that we now take for granted.

The modern view of neural networks began in the 1940s with the work of Warren McCulloch and Walter Pitts, who showed that networks of artificial neurons could, in principle, compute any arithmetic or logical function. Their work is often acknowledged as the origin of the neural network field.

The first practical application of artificial neural networks came in the late 1950s, with the invention of the perceptron network and associated learning rule by Frank Rosenblatt. Rosenblatt and his colleagues built a perceptron network and demonstrated its ability to perform pattern recognition. At about the same time, Bernard Widrow and Ted Hoff introduced a new learning algorithm and used it to train adaptive linear neural networks, which were similar in structure and capability to Rosenblatt's perceptron. The Widrow-Hoff learning rule is still in use today.

Interest in neural networks had faltered during the late 1960s because of the lack of new ideas and powerful computers with which to experiment. During the 1980s both of these impediments were overcome, and research in neural networks increased dramatically. New personal computers and workstations, which rapidly grew in capability, became widely available. In addition, important new concepts were introduced.

Two new concepts were most responsible for the rebirth of neural networks. The first was the use of statistical mechanics to explain the operation of a certain class of recurrent network, which could be used as an associative memory.

The second key development of the 1980s was the backpropagation algorithm for training multilayer perceptron networks, which was discovered independently by several different researchers.

Many of the advances in neural networks have had to do with new concepts, such as innovative architectures and training rules. Just as important has been the availability of powerful new computers on which to test these new concepts.

Neural networks have clearly taken a permanent place as important mathematical/engineering tools. They don't provide solutions to every problem, but they are essential tools to be used in appropriate situations.

1.2 Applications:

Google uses neural networks for image tagging (automatically identifying an image and assigning keywords), and Microsoft has developed neural networks that can help convert spoken English speech into spoken Chinese speech. These examples are indicative of the broad range of applications that can be found for neural networks. The applications are expanding because neural networks are good at solving problems, not just in engineering, science and mathematics, but in medicine, business, finance and literature as well. Their application to a wide variety of problems in many fields makes them very attractive. Also, faster computers and faster algorithms have made it possible to use neural networks to solve complex industrial problems that formerly required too much computation.

The following list are some of neural network applications:

1. Aerospace:

High performance aircraft autopilots, flight path simulations, aircraft control systems, autopilot enhancements, aircraft component simulations, aircraft component fault detectors.

2. Automotive:

Automobile automatic guidance systems, fuel injector control, automatic braking systems, misfire detection, virtual emission sensors, warranty activity analyzers.

3. Banking:

Check and other document readers, credit application evaluators, cash forecasting, firm classification, exchange rate forecasting, predicting loan recovery rates, measuring credit risk.

4. Defense:

Weapon steering, target tracking, object discrimination, facial recognition, new kinds of sensors, sonar, radar and image signal processing including data compression, feature extraction and noise suppression, signal/image identification.

5. Electronics:

Code sequence prediction, integrated circuit chip layout, process control, chip failure analysis, machine vision, voice synthesis, nonlinear modeling.

6. Entertainment:

Animation, special effects, market forecasting.

7. Financial:

Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, credit line use analysis, portfolio trading program, corporate financial analysis, currency price prediction.

8. Insurance:

Policy application evaluation, product optimization.

9. Manufacturing:

Manufacturing process control, product design and analysis, process and machine diagnosis, real-time particle identification, visual quality inspection systems, beer testing,

welding quality analysis, paper quality prediction, computer chip quality analysis, analysis of grinding operations, chemical product design analysis, machine maintenance analysis, project bidding, planning and management, dynamic modeling of chemical process systems.

10. Medical:

Breast cancer cell analysis, EEG and ECG analysis, prosthesis design, `timization of transplant times, hospital expense reduction, hospital quality improvement, emergency room test advisement.

11. Oil and Gas:

Exploration, smart sensors, reservoir modeling, well treatment decisions, seismic interpretation.

12. Robotics:

Trajectory control, forklift robot, manipulator controllers, vision systems, autonomous vehicles.

13. Speech:

Speech recognition, speech compression, vowel classification, text to speech synthesis.

14. Securities:

Market analysis, automatic bond rating, stock trading advisory systems.

15. Telecommunications:

Image and data compression, automated information services, real-time translation of spoken language, customer payment processing systems.

16. Transportation:

Truck brake diagnosis systems, vehicle scheduling, routing systems.

1.3 Biological Inspiration:

In this section we will briefly describe the characteristics of brain function that have inspired the development of artificial neural networks.

The brain consists of a large number of highly connected elements called neurons. These neurons have three principal components: the dendrites, the cell body and the axon as shown in Figure 1.

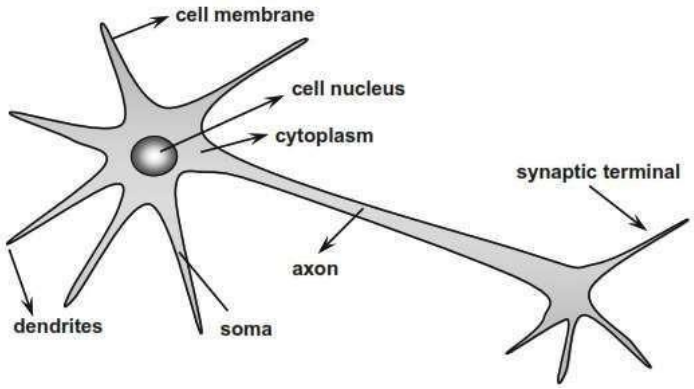

Figure 1: Biological Neuron

The dendrites are tree-like receptive networks of nerve fibers that carry electrical signals into the cell body.

The cell body effectively sums and thresholds these incoming signals.

The axon is a single long fiber that carries the signal from the cell body out to other neurons.

The point of contact between an axon of one cell and a dendrite of another cell is called a synapse. It is the arrangement of neurons and the strengths of the individual synapses, determined by a complex chemical process, that establishes the function of the neural network.

The synapses are the connections which enable the transfer of electric axon impulses from a particular neuron to dendrites of other neurons, as illustrated in Figure 2.

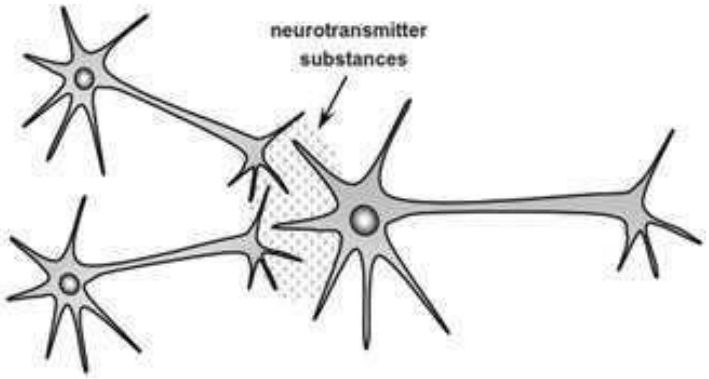

Figure 2: The synaptic connection between neurons

2. Neuron Model and Network Architectures:

Notation: Scalars — small italic letters: a, b, c
Vectors — small **bold** non italic letters: **a**, **b**, **c**
Matrices — capital **BOLD** non italic letters: **A**, **B**, **C**

1. Neuron Model:

1. Single Input Neuron:

A single-input neuron is shown in Figure 3. The scalar input p is multiplied by the scalar weight w to form, one of the terms that is sent to the summer. The other input, 1 , is multiplied by a bias (offset) b and then passed to the summer. The summer output n , often referred to as the net input, goes into a transfer function (activation function) f , which produces the scalar neuron output.

If we relate this simple model back to the biological neuron that we discussed in section 1.3, the weight w corresponds to the strength of a synapse, the cell body is represented by the summation and the transfer function, and the neuron output a represents the signal on the axon.

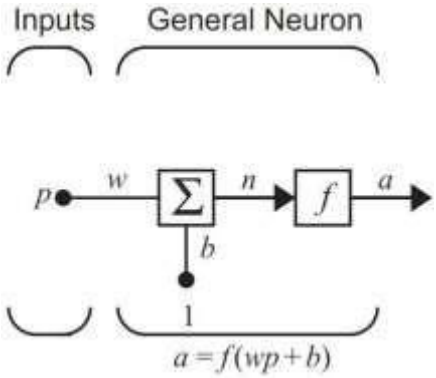

Figure 3: Single-Input Neuron

The neuron output is calculated as

$$a = f(wp + b)$$

Example 2.1:

Let $w = 3$, $p = 2$ and $b = -1.5$, what is the single-input neuron output ?

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

The actual output depends on the particular transfer function that is chosen.

Notes:

1. The bias is much like a weight, except that it has a constant input of 1. However, if you do not want to have a bias in a particular neuron, it can be omitted.
2. Note that w and b are both adjustable scalar parameters of the neuron. Typically, the transfer function is chosen by the designer and then the parameters w and b will be adjusted by some learning rule so that the neuron input/output relationship meets some specific goal.

2.1.2 Transfer Functions (Activation Functions):

The transfer function in Figure 3 may be a linear or a nonlinear function of n. A particular transfer function is chosen to satisfy some specification of the problem that the neuron is attempting to solve.

There are variety of transfer functions some of them are listed below:

1. Threshold (Hard Limit) Transfer Function:

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

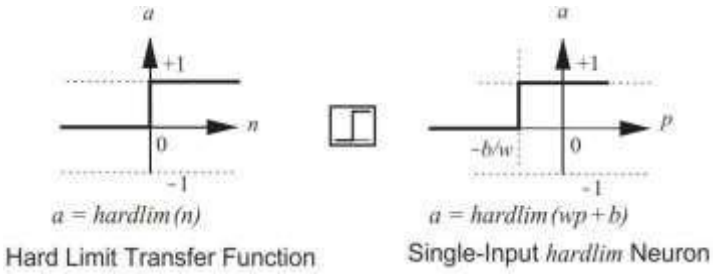

Figure 4: Hard Limit Transfer Function

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

2. Symmetric Hard Limit Transfer Function:

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

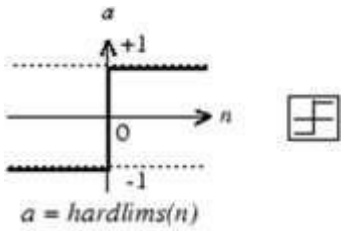

Figure 5: Symmetric Hard Limit Transfer Function

3. Linear Transfer Function:

$$f(n) = n$$

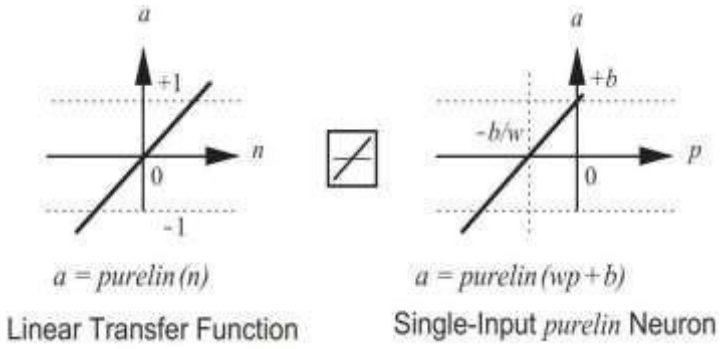

Figure 6: Linear Transfer Function

4. Logistic (Log-Sigmoid) Transfer Function:

Logistic function is a standard sigmoid function and is defined by

$$f(n) = \frac{1}{1+e^{-n}}$$

The derivative of f is defined by $f' = f\,(1 - f)$

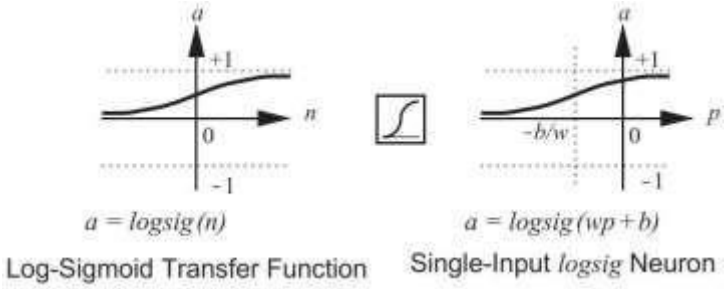

Figure 7: Log-Sigmoid Transfer Function

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

5. Hyperbolic Tangent Transfer Function:

The hyperbolic tangent is a sigmoid function and is defined by

$$f(n) = \frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$$

The derivative of f is defined by $f' = (1 - f^2)$

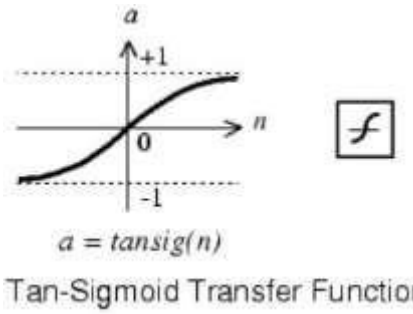

Figure 8: Hyperbolic Tangent Transfer Function

Since $\frac{\tanh^{n}(\frac{}{2})+1}{2} = \frac{1}{1+e^{-n}}$ then using the tanh function instead of the logistic one is equivalent. The tanh function has the advantage of being symmetrical with respect to the origin.

6. Radial Basis Transfer Function:

$$f(n) = e^{-n^2}$$

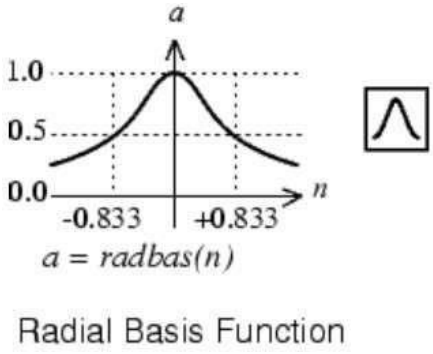

Figure 9: Radial Basis Transfer Function

7. Saturating Linear Transfer Function:

$$f(n) = \begin{cases} 0 & if \quad n < 0 \\ n & if \quad 0 \le n \le 1 \\ 1 & if \quad n > 1 \end{cases}$$

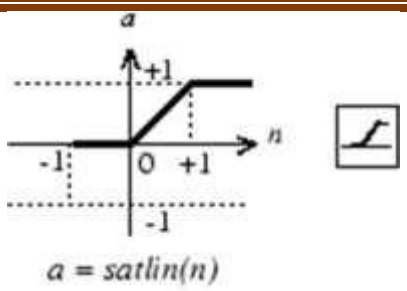

Figure 10: Saturating Transfer Function

8. Symmetric Saturating Linear Transfer Function:

$$f(n) = \begin{cases} -1 & if \quad n < -1 \\ n & if \quad -1 \le n \le 1 \\ 1 & if \quad n > 1 \end{cases}$$

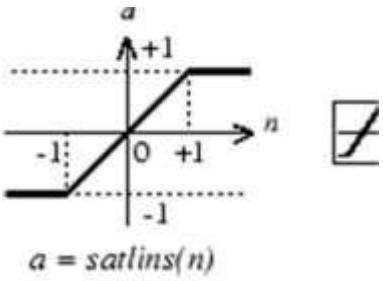

Figure 11: Symmetric Saturating Linear Transfer Function

9. Positive Linear Transfer Function:

$$f(n) = \begin{cases} 0 & if \quad n < 0 \\ n & if \quad n \ge 0 \end{cases}$$

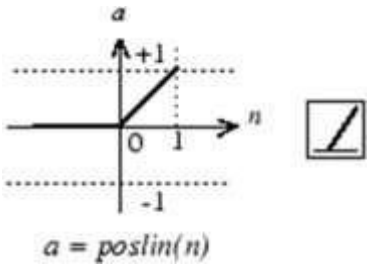

Figure 12: Positive Linear Transfer Function

Example 2.2:

Let $w = 4$, $p = 2$ and $b = -2$ with f radial basis, what is the single neuron output?

$$f = e^{-n^2}$$

$$a = e^{-(4*2+(-2))^2} = e^{-(6)^2} = 2.31952E-16$$

2.1.3 Multiple Input Neuron:

Typically, a neuron has more than one input. A neuron with R inputs is shown in Figure 13. The individual inputs $\mathbf{p} = (p_1, p_2, p_3, \ldots, p_R)$ are each weighted by corresponding elements $w_{1,1}, w_{1,2}, \ldots, w_{1,R}$ of the *weight matrix* **W**.

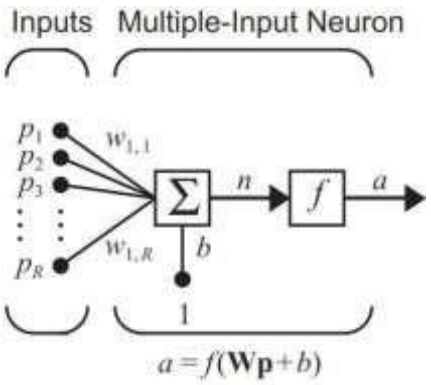

Figure 13: Multiple-Input Neuron

The neuron has a bias, which is summed with the weighted inputs to form the net input n :

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \ldots + w_{1,R}p_R + b$$

This expression can be written in matrix form:

$$n = \mathbf{W}\mathbf{p} + b$$

where the matrix **W** for the single neuron case has only one row.

Now the neuron output can be written as:

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

The elements of the weight matrix had indices, which are, the first index indicates the particular neuron destination for that weight. The second index indicates the source of the signal fed to the neuron. Thus, the indices in $w_{1,2}$ say that this weight represents the connection *to* the first (and only) neuron *from* the second source. Of course, this convention is more useful if there is more than one neuron, as will be the case later.

We would like to draw networks with several neurons, each having several inputs. Further, we would like to have more than one layer of neurons. You can imagine how complex such a network might appear if all the lines were drawn. It would take a lot of ink, could hardly be read, and the mass of detail might obscure the main features. Thus, we will use an abbreviated notation. A multiple-input neuron using this notation is shown in Figure 14.

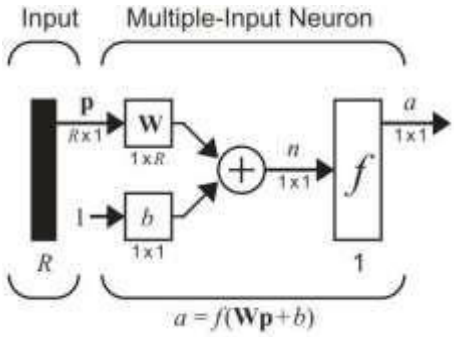

Figure 14: Neuron with *R* Inputs, Abbreviated Notation

Note that the number of inputs to a network is set by the external specifications of the problem. If, for instance, you want to design a neural network that is to predict kite-flying conditions and the inputs are air temperature, wind velocity and humidity, then there would be three inputs to the network.

2. Network Architectures:

Commonly one neuron, even with many inputs, may not be sufficient. We might need five or ten, operating in parallel, in what we will call a "layer". This concept of a layer is discussed below.

1. Single Layer network:

A single layer network of neurons is shown in Figure 15. Note that each of the inputs is connected to each of the neurons and that the weight matrix now has S rows.

The layer includes the weight matrix, the summers, the bias vector, the transfer function boxes and the output vector.

Each element of the input vector **p** is connected to each neuron through the weight matrix **W**. Each neuron has a bias b_i , a summer, a transfer function f and an output a_i . Taken together, the outputs form the output vector.

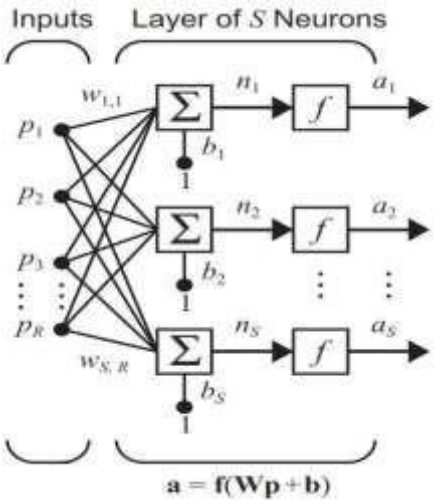

Figure 15: Single Layer of S Neurons

You might ask if all the neurons in a layer must have the same transfer function. The answer is no; you can define a single (composite) layer of neurons having different transfer