Al-Mustaqbal University

College of Engineering

Department of Biomedical Engineering

Stage: 5th Subject : Neural Eng.

2025-2026

Lecture ( ):

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.

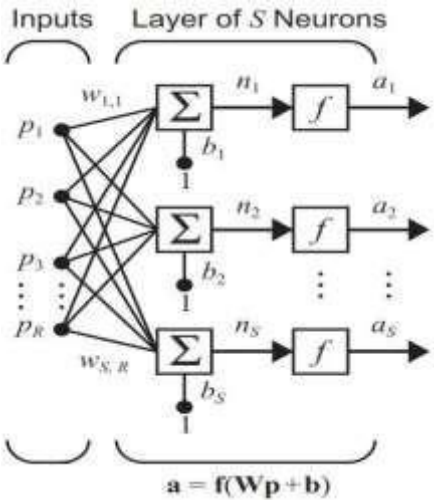

Figure 15: Single Layer of S Neurons

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

functions by combining two of the networks shown above in parallel. Both networks would have the same inputs, and each network would create some of the outputs.

The input vector elements enter the network through the weight matrix **W**:

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

no. of neurons

As noted previously, the row indices of the elements of matrix **W** indicate the destination neuron associated with that weight, while the column indices indicate the source of the input for that weight. Thus, the indices in $w_{3,2}$ say that this weight represents the connection *to* the third neuron *from* the second source.

input

Fortunately, the S-neuron, R-input, one-layer network also can be drawn in abbreviated notation, as shown in Figure 16.

Matrix of input

Matrix of neurons

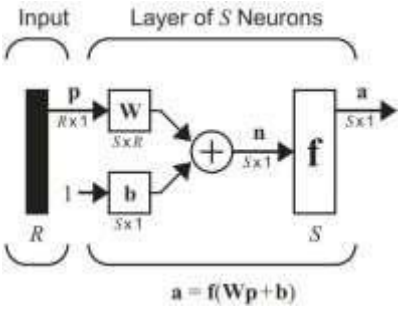

Figure 16: Layer of *S* Neurons, Abbreviated Notation

2.2.2 Multiple Layers Network:

Now consider a network with several layers. Each layer has its own weight matrix, its own bias vector **b** , a net input vector **n** and an output vector **a**. We need to introduce some additional notation to distinguish between these layers. We will use superscripts to identify the layers. Specifically, we append the number of the layer as a *superscript* to the names for each of these variables. Thus, the weight matrix for the first layer is written as $\mathbf{W}^1$, and the weight matrix for the second layer is written as $\mathbf{W}^2$. This notation is used in the three-layer network shown in Figure 17.

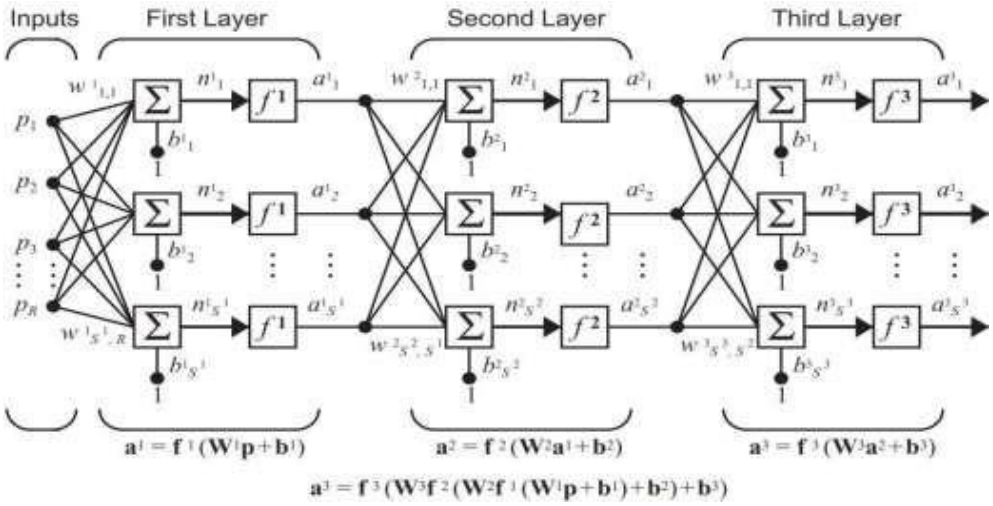

Figure 17: Three-Layer Network

As shown, there are R inputs, S^1 neurons in the first layer, S^2 neurons in the second layer, etc. As noted, different layers can have different numbers of neurons.

The outputs of layers one and two are the inputs for layers two and three. Thus layer 2 can be viewed as a one-layer network with $R = S^1$ inputs, $S = S^2$ neurons, and an $S^2 \times S^1$ weight matrix $\mathbf{W^2}$. The input to layer 2 is $\mathbf{a^1}$, and the output is $\mathbf{a^2}$.

A layer whose output is the network output is called an *output layer*. The other layers are called *hidden layers*. The network shown above has an output layer (layer 3) and two hidden layers (layers 1 and 2).

The same three-layer network discussed previously also can be drawn using our abbreviated notation, as shown in Figure 18.

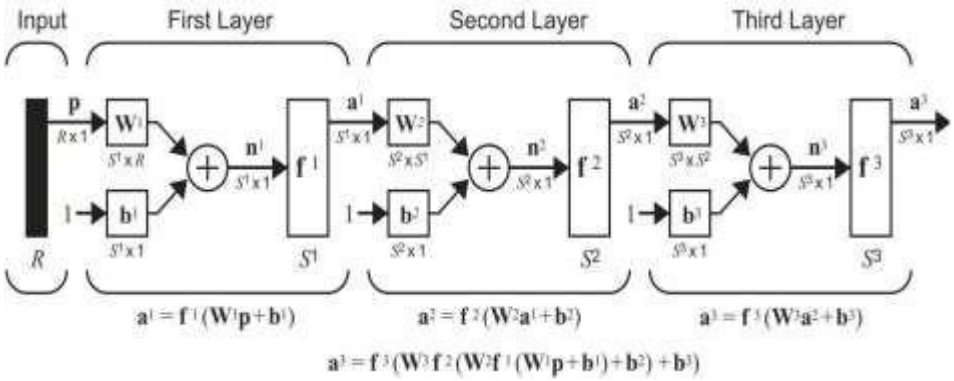

Figure 18: Three-Layer Network, Abbreviated Notation

Multilayer networks are more powerful than single layer networks. For instance, a two-layer network having a sigmoid first layer and a linear second layer can be trained to approximate most functions arbitrarily well. Single- layer networks cannot do this.

As for the number of layers, most practical neural networks have just two or three layers. Four or more layers are used rarely.

We should say something about the use of biases. One can choose neurons with or without biases. The bias gives the network an extra variable, and so you might expect that networks with biases would be more powerful than those without, and that is true. Note, for instance, that a neuron without a bias will always have a net input n of zero when the network inputs **p** are zero. This may not be desirable and can be avoided by the use of a bias.

2.2.3 Recurrent Networks:

A *recurrent network* is a network with feedback; some of its outputs are connected to its inputs. This is quite different from the networks that we have mentioned before, which were strictly feedforward with no backward connections. It is similar to feedforward neural network with no limitations regarding backloops. In these cases information is no longer transmitted only in one direction but it is also transmitted backwards. Figure 19 shows small *Fully Recurrent* artificial neural network and complexity of its artificial neuron interconnections. The most basic topology of recurrent artificial neural network is fully recurrent artificial network where every basic building block (artificial neuron) is directly connected to every other basic building block in all direction.

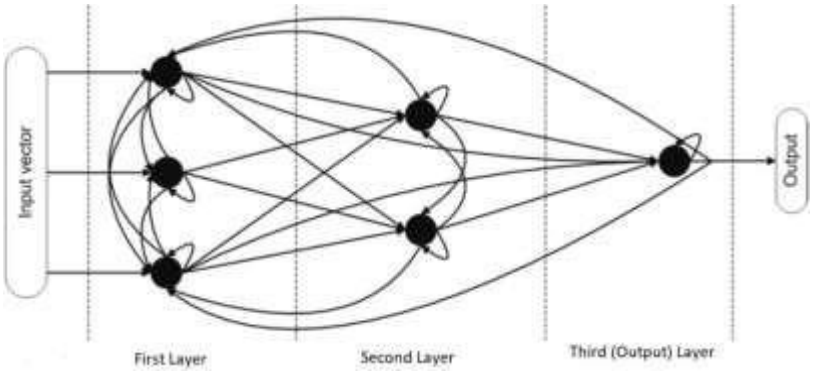

Figure 19: Fully Recurrent Artificial Neural Network

Other recurrent artificial neural networks such as Hopfield, Elman, Jordan, bi-directional and other networks are just special cases of recurrent artificial neural networks.

How to Pick an Architecture:

Problem specifications help define the network in the following ways:

1. Number of network inputs = number of problem inputs.
2. Number of neurons in output layer = number of problem outputs.
3. Output layer transfer function choice at least partly determined by problem specification of the outputs.

3. Examples:

3.1 Example:

Given a two-input neuron with the following parameters: $b = 1.2$, $\mathbf{W} = [3 \quad 2]$ and $\mathbf{p} = [-5 \quad 6]^T$, calculate the neuron output for the following transfer functions:

1. A symmetrical hard limit transfer function.
2. A saturating linear transfer function.
3. A hyperbolic tangent sigmoid transfer function.

Solution:

First calculate the net input :

$$n = \mathbf{W}\mathbf{p} + b = [3 \quad 2]\begin{bmatrix}-5\\6\end{bmatrix} + (1.2) = -1.8$$

Now find the outputs for each of the transfer functions.

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

$$a = f(-1.8) = -1$$

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

$$a = f(-1.8) = 0$$

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

$$a = f(-1.8) = -0.9468$$

3.2 Example:

A single-layer neural network is to have six inputs and two outputs. The outputs are to be limited to and continuous over the range 0 to 1. What can you tell about the network architecture? Specifically:

1. How many neurons are required?
2. What are the dimensions of the weight matrix?
3. What kind of transfer functions could be used?
4. Is a bias required?

Solution:

1. Two neurons, one for each output, are required.
2. The weight matrix has two rows corresponding to the two neurons and six columns corresponding to the six inputs. (The product **Wp** is a two-element vector).
3. The transfer functions is Logistic (Log-Sigmoid) would be most appropriate.
4. Not enough information is given to determine if a bias is required.

4. Types of Problems:

There are many different problems that can be solved with a neural network. However, neural networks are commonly used to address particular types of problems. The following four types of problem are frequently solved with neural networks:

- Classification.
- Prediction.
- Pattern recognition.
- Optimization.

1. Classification:

Classification is the process of classifying input into groups. For example, an insurance company may want to classify insurance applications into different risk categories, or an online organization may want its email system to classify incoming mail into groups of spam and non-spam messages.

Often, the neural network is trained by presenting it with a sample group of data and instructions as to which group each data element belongs. This allows the neural network to learn the characteristics that may indicate group membership.

2. Prediction

Prediction is another common application for neural networks. Given a time-based series of input data, a neural network will predict future values. The accuracy of the guess will be dependent upon many factors, such as the quantity and relevancy of the input data. For example, neural networks are commonly applied to problems involving predicting movements in financial markets.

3. Pattern Recognition

Pattern recognition is one of the most common uses for neural networks. Pattern recognition is a form of classification. Pattern recognition is simply the ability to recognize