CHAPTER 2

ARTIFICIAL NEURAL NETWORKS

This chapter presents some basic concepts of artificial neural networks(ANNs), starting with a biological neuron, and its modeling by an artificial neuron. Three basic activation or transfer functions are then described. A multi–layer feedforward neural network can be formed by organizing a large number of single artificial neurons in layers. Such a network can be trained for use in system modeling and control.

2.1 INTRODUCTION

Modern computers can perform a variety of well–defined tasks such as inversion of large matrices or solving differential equation systems at speeds unmatched by humans. On the other hand, man–made machines can't solve many other problems to our satisfaction yet, even though these problems can be easily disentangled by the cognitive power of humans. The brain is more effective than the digital computer when it comes to solving problems that are ill–defined and that need massive processing. To recognize a face in a crowd, the human brain uses a gigantic web of interconnected processing elements called neurons to process information from sense organs. The human brain accomplishes massive parallel information processing with

millions and even billions of neurons in parts of the brain working together. The vast processing power inherent in biological neural structures has inspired the study of the structure itself for hints on organizing man–made computer structures. Artificial neural networks(ANNs) are an attempt to imitate the biological paradigm, our brain, in structure and in function. Their creation is aimed to technically realize capabilities and characteristics such as self–organization, learning and associative memory. To achieve this, a large number of simple processor elements(PEs) are interconnected with uni–directional signal channels to single– or multi–layer networks. All these PEs work in parallel for sequential arithmetic and/or symbolic information processing. As solid state operational amplifiers and logic gates can perform computations many times faster than the biological neurons in the brain[2–1,2–2], we may expect to build relatively inexpensive machines with the capability to process as much information as the human brain.

Definition 2.1: An artificial neural network is a massively connected processor that has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects[2–1]; (1) knowledge is acquired by the network through a learning process; and (2) interneuron connection strengths known as synaptic weights are used to store the knowledge.

The primary similarity between biological nervous systems and artificial neural networks is that each of the two consists of a large number of simple

elements that can learn and are able to collectively solve complicated and ambiguous problems[2–3]. Today, most research on ANNs is accomp–lished by simulating these networks on serial computers. Speed limitations have kept such networks relatively small. However, their applications are versatile. Apart from those reviewed in Chapter 1, more general applications include truck control system[2–4], robot control[2–5], and medical diagnosis[2–6]. Other excellent references are listed as [2–7]~[2–10].

In this chapter, the fundamental working principle of a biological neuron and its modeling by an artificial neuron as a processing element(PE) is presented. When a large number of such PEs are organized topologically to form a network, it will possess the ability of learning, adaptation and generalization.

2.2 A BIOLOGICAL NEURON

Neural networks are aggregates of interconnected nerve cells. A nerve cell comprises the cell body(soma) which surrounds the cell nucleus. The cell body has a long processus, i.e., the axon which ends in numerous ramifications attached to other cells via so–called synaptic end–heads thus forming the synaptic connections between neurons[2–2]. A branching structure called dentite picks up signals from other neurons. At the tail end of the axon, these are brushlike structures called synaptic buttons, as shown in Figure 2.1.

In the stationary electrochemical state the cell has a resting potential of about –80mV. If a cell is stimulated by another cell via the synaptic connection, a pole reversal of about 30mV results in a very short period of time. This action or activation potential moves across the axon to neighboring cells with a speed of up to 100 meters per second. To generate this action potential, the stimulus must exceed a specific threshold. The activation potential drops immediately after its rise and the cell returns to the original state, as shown in Figure 2.2. The reaction of a neuron to a stimulus received depends on its history, for instance, how many impulses it has transmitted before and in which sequence. This is the learning and adaptive capability of the biological neural networks, or the plasticity characteristic of the synapses which gives the neurons a memory.

2.3 MATHEMATICAL MODEL OF A NEURON

On the working principle of the biological neuron, a mathematical model can be constructed as shown in Figure 2.3, where θ_k is the threshold value, h_k the inhibitory input. The neuron acts like an integrator with feedback. The integrator integrates the weighted presynaptic input signals and maps the postsynaptic activation x_j on to the output signal by means of a nonlinear function. The differential equation for the activation u_k describes the short–term behavior of the neuron as a function of the input signal x_i , i.e.,

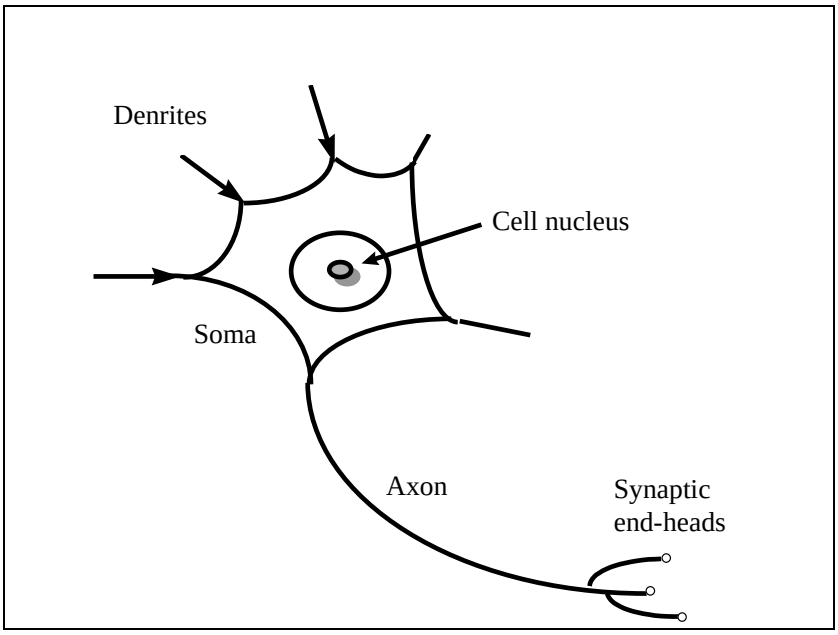

Fig. 2.1 A Biological Neuron

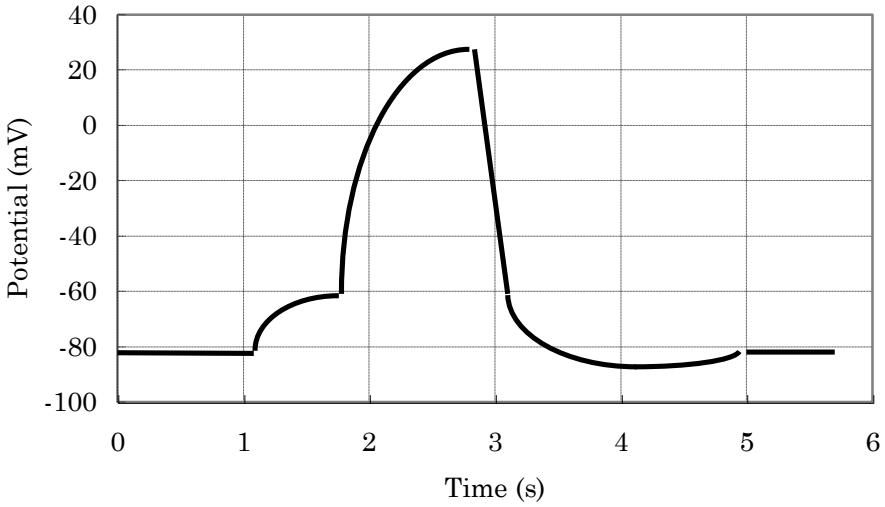

Fig. 2.2 Response to A Stimulus

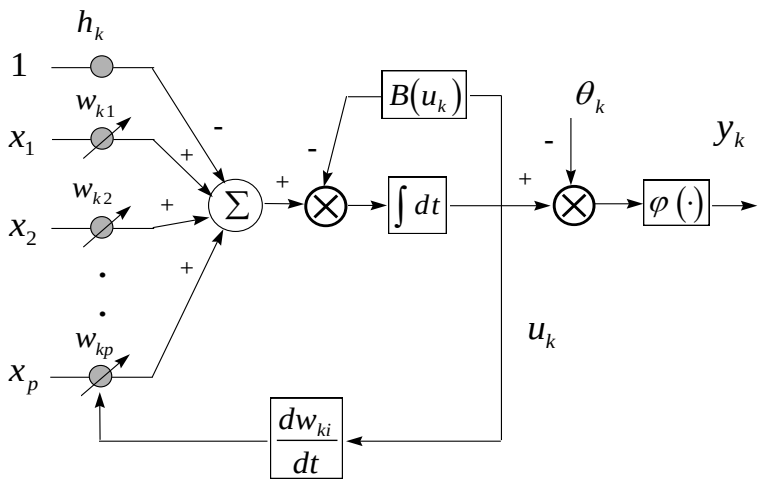

Fig. 2.3 Mathematical Model of the Neuron

$$\frac{du_k}{dt} = -B(u_k) + \sum_{i=1}^{n} w_{ki} x_i - h_k \tag{2.1}$$

To model the plasticity property of the biological neuron, the following differential equation is used to adapt the connecting weights, i.e.,

$$\frac{dw_{ki}}{dt} = \eta(t) \cdot \varphi(x_i, u_k, w_{ki}) \tag{2.2}$$

It describes the dynamics of learning as a function of the instantaneous values of the connecting weights, the activation and the input quality. $\eta(t)$ is a learning factor governing the adapting speed of the weights.

A further simplified model can be obtained as shown in Figure 2.4. It was the earliest contribution using such a model to simulate the way neural nets work. The output of the model can be expressed as

$$y_k = \varphi(u_k - \theta_k) = \varphi\left(\sum_{j=1}^{p} w_{kj} x_j - \theta_k\right) \tag{2.3}$$

where

x_i = the input signals,

w_{ki} = the synaptic weights of neuron k;

u_k = the output of the linear combiner;

θ_k = the threshold;

φ = an activation function; and

y_k = the output signal of the neuron.

The threshold θ_k can be both positive and negative. When it is positive, it has the effect of lowering the net input of the activation function. When it is negative, it has the effect of increasing the net input of the activation function. In the latter case, the threshold becomes a bias which is the negative of the threshold. Replacing θ_k in (2.3) by a weight w_k with a constant input x_0 fixed at either -1 or $+1$, corresponding to θ_k being negative or positive, respectively, the final model of the neuron is given by Figure 2.5. Biological neurons are living cells capable of receiving and transmitting electrochemical signals in highly specialized ways. Their complexities can be accurately simulated only by intricate computer chips. The modeling of the biological neurons by artificial neural nets are extremely simplified representations of the real thing.

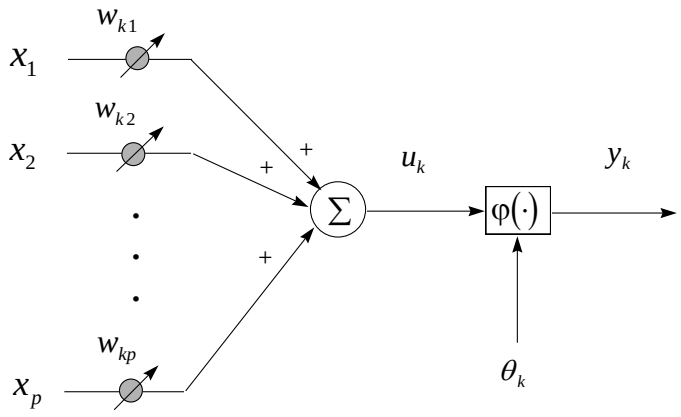

Fig. 2.4 Nonlinear Model of A Neuron

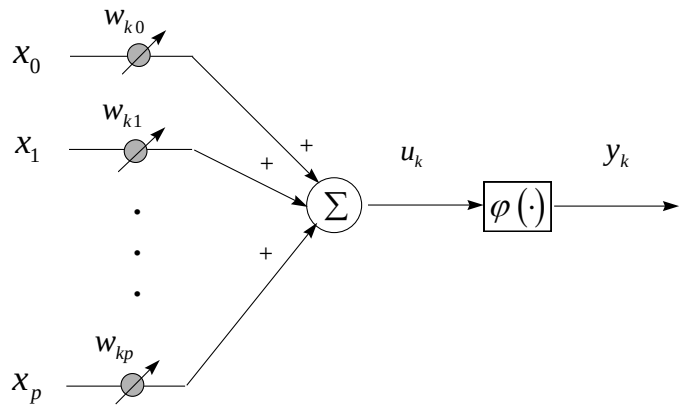

Fig. 2.5 Final Model of the Neuron

2.4 ACTUATION FUNCTIONS

To describe the output, y, of a neuron in terms of the net activity level, x, numerous nonlinear transfer functions can be utilized. We present three of the most commonly used transfer functions here. For a more comprehensive listing and description of these and other functions, please refer to[2–1,2–10].

(1) *Symmetric hard Limit Transfer Function*

Shown in Figure 2.6, this transfer function forces a neuron to produce an output of +1 if its net input reaches a threshold, otherwise a –1. It is often used in a network for decision making or classification. The mathematical expression is given by

$$y = \begin{cases} +1, & x \geq 0, \\ -1, & x < 0. \end{cases} \tag{2.4}$$

(2) *Tan Sigmoid Transfer Function*

Shown in Figure 2.7, this transfer function provides saturation for decision making, yet they have differentiable input–output characteristics that facilitate adaptivity. It maps the input of a neuron from the interval $(-\infty,+\infty)$ to the interval $(-1,+1)$. The mathematical expression is given by

$$y = \tanh\left(\frac{1-e^{-2x}}{1+e^{-2x}}\right) \tag{2.5}$$

(3) *Log Sigmoid Transfer Function*

Shown in Figure 2.8, this squashing transfer function can be viewed as performing a function similar to an analog electronic amplifier. The gain of the amplifier is analogous to the ratio of the change in output for a given change in input. When the inputs are near zero, the gain is greatest. This will mitigate problems associated with possible dominating effects of large input signals. The mathematical expression of the transfer function is given as follows:

$$y = \frac{1}{1+e^{-x}} \tag{2.6}$$

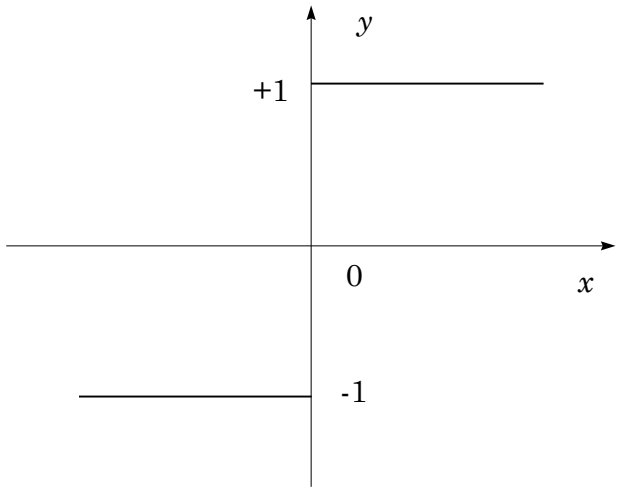

Fig. 2.6 Symmetric Hard Limit Activation Function

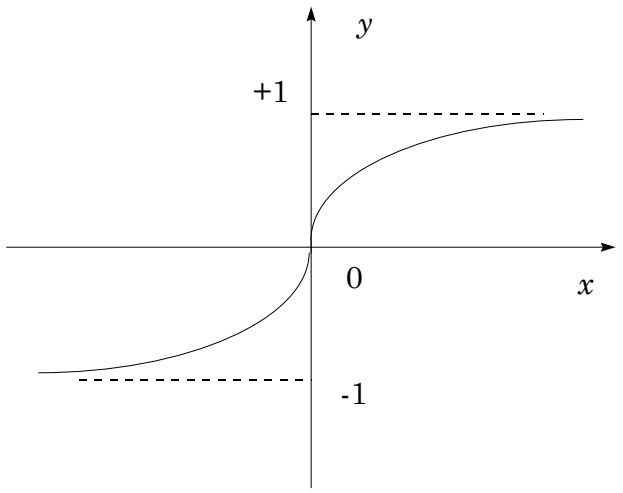

Fig. 2.7 Tanh Sigmoid Activation Function

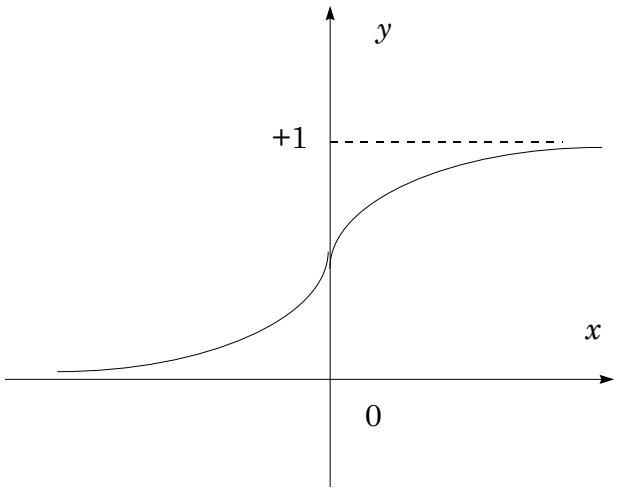

Fig. 2.8 Log Sigmoid Activation Function

2.5 NEURAL NETWORK ARCHITECTURES

An artificial neuron described by the above model is extremely simple. For solving complex problems, a large number of such simple processing elements(PEs) can be organized topologically to form different structures of networks. The combined characteristic features of parallel processing, adaptive connection weights, and nonlinear output functions yield the possibility for the design of neural networks with unprecedented computing power. There are different ways to categorize network architectures. Depending on the mathematical equations used to describe the model, two types of architectures can be identified, i.e., one by nonlinear differential equations and the other by algebraic expressions. According to the actual structure of the network, four classes of network architectures can be identified[2–1]. They are:

- single–layer feedforward networks;
- multilayer feedforward networks;
- recurrent networks and
- lattice structures.

In this thesis, only multilayer feedforward networks are used. Therefore, the structure of such networks is discussed in more detail. A layered neural network is one whose neurons are organized in the form of layers. Figure 2.9 shows a typical multilayer feedforward network. Because the signals flow

only in one direction, i.e., from left to right, the network is called feedforward. On the leftmost is an input layer. On the rightmost is an output layer. The layers in between are called hidden layers. The neurons in hidden layers are called hidden neurons or hidden units. This network is said to be fully connected as every node in each layer is connected to every other node in the adjacent forward layer. If some of these connections are missing, such a network is called a partially connected network.

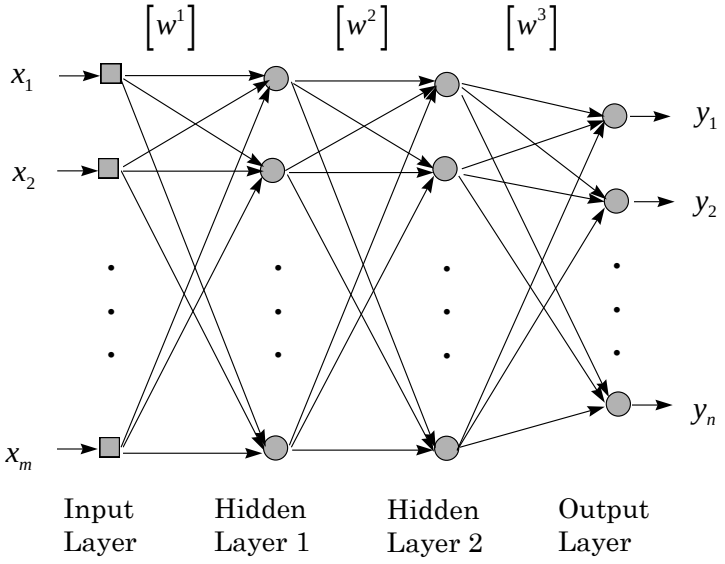

Fig. 2.9 A Feedforward Multilayer Neural Network

2.6 LEARNING RULES

In designing a neural network for modeling a process or a plant, one important step is to teach the network the environment. The network improves its performance through learning. The realization of this learning is

achieved through an iterative process of adjustments of the synaptic weights and thresholds. The learning process is complicated and hard to define with precision.

Definition 2.2: Learning is a process by which the free parameters of a neural network are adaptive through a continuing process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameters adjustment takes place.

There are many learning rules such as error–correction learning rooted from optimum filtering, Heblian learning and competitive learning inspired by neurobiological considerations, and Boltsman learning borrowed from thermodynamics and information theory. On the other hand, a model of the environment in which the designed neural network operates is referred to as a learning paradigm. There are three learning paradigms, i.e., supervised learning, reinforcement learning by use of critic, and self–organized learning which is also called unsupervised learning where no teacher or critic is required. In this thesis, supervised learning is used. This subject will be further covered in Chapter 4.

2.7 MULTILAYER PERCEPTRONS

In this section, the detailed derivation of the delta training rule for multilayer perceptrons, i.e., multilayer feedforward neural networks is presented. Such networks can be successfully trained using the supervised

paradigm called back–propagation algorithm based on error–correction learning.

The training process of the algorithm consists of two passes of a signal through the network. In the forward pass, input data are applied to the input layer as stimuli and are propagated to the output neurons where a set of outputs are obtained and compared with the teacher's output values. An error vector is then produced. In the backward pass, the error vector is backward propagated to adjust the synaptic weights. This process continues until the norm of the error vector is brought to a satisfactory small number. From Figure 2.10, the error of output neuron j at iteration k can be expressed as

$$e_j^k = d_j^k - y_j^k \tag{2.7}$$

where d_j^k is the desired output of neuron j at iteration k.

The total squared error of the output layer is given by

$$E^k = \frac{1}{2}\sum_{j=1}^{N_o}\left(e_j^k\right)^2 \tag{2.8}$$

where N_o is the total number of neurons at the output layer.

If there are N_p patterns of training data, then the average squared error

$$E = \frac{1}{N_p}\sum_{k=1}^{N_p} E^k = \frac{1}{2N_p}\sum_{k=1}^{N_p}\sum_{j=1}^{N_o}\left(e_j^k\right)^2 \tag{2.9}$$

The goal when training a neural network is to minimize E by adjusting the synaptic weights. In Figure 2.10, the net internal activation of neuron j at iteration k, v_j^k, is given by

$$v_j^k = \sum_{i=0}^{N_m} w_{ji}^k y_i^k \tag{2.10}$$

where

N_m = the number of neurons at the $m-$th hidden layer,

$w_{j0} = \theta_j$, the threshold; and

$y_0 = -1$.

Therefore, the output of output neuron j at iteration k is given by

$$y_j^k = \varphi\left(v_j^k\right) \tag{2.11}$$

Now our task is to find out how to modify w_{ji}, knowing everything else. Using the chain rule in calculus, we have

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial e_j}\frac{\partial e_j}{\partial y_j}\frac{\partial y_j}{\partial v_j}\frac{\partial v_j}{\partial w_{ji}} \tag{2.12}$$

where the superscript k in the formula is omitted.

Carrying out the derivatives, we can have

$$\frac{\partial E^k}{\partial w_{ji}^k} = -e_j^k \cdot \varphi'\left(v_j^k\right) \cdot y_j^k \tag{2.13}$$

The correction of w_{ji}^k can then be made by

$$w_{ji}^{k+1} = w_{ji}^k + \Delta w_{ji}^k \tag{2.14}$$

where

$\Delta w_{ji}^{k} = -\eta \cdot \frac{\partial E^{k}}{\partial w_{ji}^{k}}$, called delta rule, and

η = a learning rate that controls the learning speed.

For a hidden neuron j, the above formula has to be modified. Please refer to [2–1] for details.

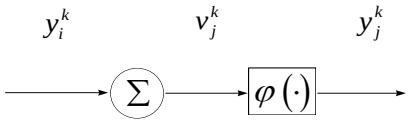

Fig. 2.10 Output Neuron j