Chapter 5

INTRODUCTION TO ARTIFICIAL NEURAL NETWORK

Artificial neural networks or simply “neural nets” go by many names such as connectionist models, parallel distributed processing models, and neuromorphic systems. Whatever terminology it may be, they all attempt to borrow the structure and running way of the biological nervous system based on our present understanding of it. Instead of performing a program consisting of instructions sequentially as in a von Neumann computer, artificial neural nets have their structures in dense interconnection of simple computational elements—the artificial neurons or simply “neurons”, and operate the massive computational elements in parallel to achieve high performance speed.

Neurons in a neural net can be viewed as nodes in a layer network defined in Section 1.5, but as a node in a neural net the neuron not only sums up the weighted inputs from other nodes in one of the neighboring layers but also performs a nonlinear transformation on the summation, then the output of this neuron will be sent to all neurons in the next layer with links to it (in a network in graph theory a node distributes its received input to nodes linked). Nodes or computational elements in neural nets are nonlinear and typically realized by analog circuits. Different types of nodes, distinguished by types of nonlinearities, can be used in one network. So there are three key factors for specifying a neural net:

- The net topology. Topological factors include: feedforward type network or feedback type network, the number of layers, and the number of nodes in each layer.

- The type of nodes (neurons). Different nonlinearities realized by analog circuits or more complex mathematical operations realized by digital circuitries can be considered. The type of neurons also determines the time
feature of the network operation: the nodes operate continuously or at discrete amounts of time.

- The weights specification. There are two cases: predetermined weights and adapted weights. Adaptation or learning is the main feature of artificial neural nets. The ability to adapt and continue learning is essential in areas such as speech recognition.

Classification of neural nets then can be made on the basis of the above three factors. Considering the topological structure, there are two classes of neural nets: *feedforward neural nets* and *feedback neural nets*. Nets introduced by Hopfield ([148], [150], [152]) are in the class of feedback nets. A class of artificial neural networks called *perceptrons* ([261], [225]) introduced by Rosenblatt has the feedforward structure.

Different nonlinearities of the neurons specify the features of input and output of the nets. A *binary net* is a net with binary input and output. These nets are most appropriate in processing images which have pixel values 1 or −1 (black and white image). A *continuous net* is a net with continuous input and output. There could be hybrid nets which have binary input and continuous output, or continuous input and binary output.

Training strategy is also a key to classify the artificial neural nets. An *adaptive net* is a net whose weights are trained during the operation by learning or self-adaptation. Nets trained by self-organization or self-adaptation (self-organized nets) are also said to be nets trained without supervision. As we will see, Kohonen's feature-map forming nets ([180], [183]) are typical nets without supervision. Nets trained by learning are said to be nets trained with supervision. Nets with fixed weights are *non-adaptive nets*, and most of the artificial neural nets that were designed for solving mathematical programming problems are non-adaptive nets as we will see in the following chapters.

### 5.1 WHAT IS AN ARTIFICIAL NEURON?

Mathematically, an artificial neuron is a composite nonlinear function \( a(x) : \mathbb{R}^n \rightarrow \mathbb{R} \) (shortly we call \( a(x) \) the *neuron function*). More precisely,

\[
a(x) = \phi(\sigma(x) - T),
\]

where \( T \in \mathbb{R} \) is the *external threshold* or simply *threshold*, \( \sigma(x) : \mathbb{R}^n \rightarrow \mathbb{R} \) is called an accumulation function. And \( \phi(x) : \mathbb{R} \rightarrow \mathbb{R} \) is a nonlinear function called *activation function*. The accumulation function is generally a linear function of the input \( x \). In this case,

\[
a(x) = \phi(w^T x - T),
\]
which implies that the types of $\phi(x)$ specify the features of the neuron and furthermore the whole network. Fig. 5.1 shows the general structure of a neuron with linear accumulation function. A neuron with activation mechanism of (5.2) is also called a threshold-activated neuron.

The following neuron models have played important roles in the development of artificial neural nets:

- **McCulloch-Pitts neuron model**[219][12]

  In McCulloch-Pitts model the activation function $\phi(x)$ is a *unipolar binary function* as follows: (see Fig. 5.2).

$$
\phi(\sigma(x) - T) = \begin{cases} 
1, & \sigma(x) - T \geq 0 \\
0, & \sigma(x) - T < 0,
\end{cases}
$$

(5.3)

where the weighted sum of the inputs, $\sigma(x)$, is compared with the threshold $T$. If this sum exceeds the threshold, the neuron output is set to the “high level” 1, otherwise to the “low level” 0. In the analog theory, it is commonly referred to as the *hard-limiting function* or *hard limiter*.

In this model the accumulation function is a linear function:

$$
\sigma(x) = \sum_{i=1}^{n} w_i x_i,
$$

(5.4)

where $w_i$ is referred to as *synaptic weight* or *link weight*, or *weight* for short, which describes the connection relationship between this neuron and the neuron that gives the input $x_i$. A synaptic weight is positive or nonpositive if the connection is “exciting” or “inhibitory”.

---

**Figure 5.1.** General structure of a neuron with linear accumulation function
A variation of the McCulloch-Pitts neuron model, called the signum neuron, has its binary activation function (bipolar binary function) as:

\[
\phi(\sigma(x) - T) = \text{sgn}(\sigma(x) - T) = \begin{cases} 
1, & \sigma(x) - T \geq 0 \\
-1, & \sigma(x) - T < 0.
\end{cases}
\] (5.5)

- Fukushima neuron model[108],[109]

In Fukushima model the activation function \( \phi(x) \) is the so-called simple limiter function (see Fig. 5.3):

\[
\phi(\sigma(x) - 1) = \begin{cases} 
\sigma(x) - 1, & \sigma(x) - 1 \geq 0 \\
0, & \sigma(x) - 1 < 0,
\end{cases}
\] (5.6)

with the threshold here being fixed at 1. The accumulation function is a linear fractional function. Let \( x^T = (x^eT, x^hT) \), \( x^e \in \mathbb{R}^{n_1}, x^h \in \mathbb{R}^{n_2} \), \( n_1 + n_2 = n \). \( x^e \) is a group of neurons which have exciting connections with the current neuron and \( x^h \) is an inhibitory group to the current neuron.
The Fukushima neuron with simple limiters

\[ y = \begin{cases} 
\sigma(x) - 1 & \text{if } \sigma(x) - 1 \geq 0 \\
0 & \text{otherwise}
\end{cases} \]

where all the weights are nonnegative. Fukushima neurons are used in a class of neural network structures known as neocognitrons (see [109], [110]) which is a model for visual pattern recognitrons and is concerned with biological plausibility.

- **Sigmoidal neuron**

In the Hopfield model ([150], [151], [280], or see the next chapter), there is a class of analog neurons called amplifier with saturation property (see Fig. 5.4). Its neuron function is the sigmoidal function:

\[ \sigma(x) = \frac{1 + \sum_{i=1}^{n_1} w_i^e x_i^e}{1 + \sum_{i=1}^{n_2} w_i^h x_i^h}, \quad (5.7) \]

with 1 and \(-1\) being two polar values (bipolar continuous function). So we refer to the amplifier as a sigmoidal neuron. As we will see below,
sigmoidal neuron plays an important role in the perceptron model ([12], [261], or see the next chapter).

The following sigmoidal function with polar values 1 and 0,

\[ \phi_u(x) = \frac{1}{2} \left( 1 + \tanh(\lambda x) \right) = \frac{1}{1 + e^{-2\lambda x}}, \]  

(5.9)

is called a unipolar continuous function, where \( \lambda \) is a positive constant which controls the "steepness" of the sigmoidal function. For a sufficiently large \( \lambda \), the sigmoidal function is an approximation to the hard limiter. In the Hopfield neuron model, the accumulation function is linear as in (5.4).

The sigmoidal activation function is differentiable, whose derivative with respect to \( x \) is

\[ \dot{\phi}_b(x) = \lambda (1 - \phi_b^2(x)), \]  

(5.10)

\[ \ddot{\phi}_b(x) = -2\lambda \dot{\phi}_b(x) \dot{\phi}_b(x) \]  

\[ = -\lambda^2 \phi_b(x)(1 - \phi_b^2(x)) \]  

(5.11)

or

\[ \dot{\phi}_u(x) = 2\lambda \phi_u(x)(1 - \phi_u(x)), \]  

(5.12)
\[
\dot{\phi}_u(x) = 2\lambda (\dot{\phi}_u(x) - 2\phi_u(x)\dot{\phi}_u(x)) \\
= 4\lambda^2 \phi_u(x)(1 - \phi_u(x))(1 - 2\phi_u(x)).
\] (5.13)

**Higher-order neuron**\textsuperscript{[262]}

This is a class of neurons that have nonlinear accumulation functions, or more precisely, polynomial accumulation functions:

\[
\sigma(x) = \sum_{i=1}^{n} w_i x_i + \sum_{i,j=1}^{n} w_{ij} x_i x_j + \sum_{i,j,k=1}^{n} w_{ijk} x_i x_j x_k + \cdots.
\] (5.14)

Higher-order neurons are employed in invariant perceptrons (see [262], [119]) that are capable of invariant pattern recognition.

**Time-depending neuron/ the integrator neuron,**

**Grossberg model**

The accumulation functions of the neurons discussed so far have current inputs \(x(t)\) as the variables. A neuron, whose accumulation function depends not only on \(x(t)\) but some historical data \(x(\tau)\), where \(\tau \in [t - \Delta t, t]\), is referred to as a time-depending neuron, where \(\Delta t\) is a time interval. In general, its accumulation function can be expressed as \(\sigma(x(\tau), \tau \in T \subset [t - \Delta t, t])\). Consider a special case: let the threshold \(T = 0\) and the activation function \(\phi(y)\) be an identity function, and the accumulation function be

\[
\sigma(x(t), x(t - \Delta t)) = \frac{x(t)}{\Delta t} - \frac{x(t - \Delta t)}{\Delta t};
\]

then the neuron function is

\[
a(x(t)) = \phi(\sigma(x(t), x(t - \Delta t)) = \frac{x(t) - x(t - \Delta t)}{\Delta t}.\] (5.15)

If \(\Delta t \rightarrow 0\), \(a(x(t)) \rightarrow \frac{dx}{dt}\). So the integrator can be viewed as a special time-depending neuron (we call it integrator neuron). Integrator neurons are frequently employed in artificial neural network for solving optimization problems. The optimization neural networks introduced by Hopfield and Tank ([150], [280]) consist of integrator neurons as well as sigmoidal neurons as we will see in the following chapter. Recently it was
reported (see [325]) that the amplifiers with saturation property (i.e., the sigmoidal neuron, see Fig. 5.4) in the Hopfield network can be replaced by an integrator neuron with saturation property:

\[ \dot{x}_i = u_i, \quad -1 \leq x_i \leq 1 \]

where \( x_i, u_i \) are defined in Fig. 5.4. The new network with the replacement works well according to computer simulations. Their discovery gives an insight into the similarity between the sigmoidal neuron and the integrator neuron (see the details in Section 7.3).

Another typical neuron model with integrator (refer to Fig. 5.5) is presented by Grossberg ([132], [50], [51]):

\[
\begin{align*}
\dot{x}_j &= -\alpha_j x_j + (\gamma_j - \beta_j x_j) \sum_{i=1}^{n} w_{ij} y_i + \theta_j \\
y_j &= \psi(x_j), \quad j = 1, 2, \ldots, n.
\end{align*}
\]

(5.16)

Here \( y_j \) is the output, \( x_j \) is the intermediate state, \( \alpha_j, \beta_j, \gamma_j, \theta_j \) are constants responsible for forgetting, the automatic gain control, and the total activity normalization and an external input. \( w_{ij} \) are synaptic weights imposed on the inputs \( y_i \). Function \( \psi(x) \) represents a nonlinear activation.

### 5.2 FEEDFORWARD AND FEEDBACK STRUCTURES

A feedforward neural network has a structure similar to that of the weighted layer network defined in Section 1.5 except that the nodes (neurons) in a neural network have more functions than the nodes in general networks. Each neuron sums up the weighted inputs first and then applies a nonlinear transformation
on the result to produce an output, which is sent forward along each leaving arc.

In a feedforward neural network, layers with neurons whose outputs are not directly accessible are called internal or hidden layers. Sometimes one uses neurons with identity transformation functions to represent the inputs; in this case, an \(N\)-layers feedforward network has one input layer, one output layer and \(N - 2\) hidden layers.

Feedforward neural networks are employed in several applications:

- **Pattern classification** A pattern is the quantitative description of an object, event, phenomenon or process. The task of pattern classification is to assign an object, event, or phenomenon to one of the specified classes. A neural network that fulfills the task of pattern classification is called a pattern classifier. The general pattern classification problem can be posed as the following mathematical problem:

**Definition 5.1** Let \(\mathbb{R}^n\) be the pattern space and \(\{+1, -1\}^m\) be the decision space. The sets in \(\mathbb{R}^n\) containing patterns of classes \(1, 2, \ldots, C(\leq 2^m)\) are denoted by \(K_1, K_2, \ldots, K_C\) respectively. For a pattern \(x \in K_i\), the classifier maps it into one of the subsets of \(\{+1, -1\}^m\) that represents the set \(K_i\).

- **Function approximation and nonlinear signal prediction** The term function (or mapping) here represents a relationship between \(\mathbb{R}^n\) and \(\mathbb{R}^m\) for which sometimes we have no analytical expression, or, if there is an analytical expression it is too complicated to be computed for an on-line control. The general function approximation problem in terms of neural network is defined as follows:

**Definition 5.2** Let \((f_1(x), \ldots, f_m(x))\) be a mapping from \(\Omega \subset \mathbb{R}^n\) to \((\mathbb{R}^m)\). Given \(p\) sets of sampling data \((x^\mu, f_1(x^\mu), \ldots, f_m(x^\mu))\), \(\mu = 1, \ldots, p\), construct a \(k\)-layer feedforward neural network \(N_{k,w}\) with \(m\) output neurons \(y_\rho(x), \rho = 1, \ldots, m\), such that for any specified small positive value \(\varepsilon\), there exist weight values \(w\) satisfying,

\[
\sum_{\mu=1}^{p} \sum_{\rho=1}^{m} \|f_\rho(x^\mu) - y_\rho(x^\mu)\|_s \leq \varepsilon
\]

where \(s = 1\) or \(2\) or \(\infty\).

Function approximation in some cases is also referred to as adaptive modeling or system identification (see [312]). For a physical dynamic system it
may be regarded as an unknown "black box" having inputs and outputs. A feedforward neural network can be used in modeling (that is, imitating the behavior of the system with the goal of yielding its output to match that of the unknown system), generally to cause its output to be a best least-squares approximation to that of the unknown system.

Closely related to function approximation is the application of the feedforward neural networks for nonlinear signal prediction and forecasting.

Suppose that we are given a time series \( \eta_k, k = 1, 2, \ldots, K, \eta_k \in \mathbb{R} \) which is observed from a chaotic dynamical system. We want to predict future outputs \( \eta_{k_0+1}, \ldots, \eta_{k_0+m} \) of the system by using the observed data \( \eta_{k_0-p}, \ldots, \eta_{k_0} \). According to Definition 5.2, we define a mapping:

\[
f_i(\eta_{k_0-p}, \ldots, \eta_{k_0}) = \eta_{k_0+i}, \quad i = 1, \ldots, m,
\]

where \( m + p << K \). The forecasting problem now can be transformed into an approximation problem. Constructing a network mentioned in Definition 5.2 and training it by the data structure described in (5.17) with \( k_0 = p + 1, \ldots, K - m \), one can find the proper parameters for the network. To predict, using a set of \( p + 1 \) observed values the network (the mapping) will output the next \( m \) values of the time series. A number of different network architectures have been applied to the problem of predicting chaotic time series (see [95], [194]). In contrast to the conventional prediction methods such as the Linear Predictive Methods ([311], [312], [210]), the Gabor-Volterra-Weiner Polynomial Method ([113], [264]), the neural network increases the prediction accuracy and is parsimonious in its requirement for data samples from the time series, as reported by [194]. For their powerful potentials in function approximation and nonlinear signal prediction the feedforward networks have become useful tools in statistical researches ([57]).

A neural network is called feedback if output of any neuron is an input of another neuron of the same network, see Fig.5.6. So far only the simplest feedback structure has been used to construct feedback neural networks, i.e., the outputs, which in most cases are directly accessible, of the neurons at the last layer are fed back to the first layer of the network.

Artificial neural networks with feedback structures have appeared in many applications:

- **Associative memory or content-addressable memory (CAM)** Consider a neural network model in which the stored patterns are a series of \( M \)
memory vectors $y^\mu$, $\mu = 1, \ldots, M$, of dimension $N$. The memory vector represents a binary pattern, so each neuron takes $+1$ or $-1$ as state value. A CAM is defined as a memory in which the stored patterns are recalled by presenting the memory with a partial form (or a distorted form) of the stored patterns. CAM model is used in pattern recognition or pattern identification (see [144], [149], [183], [231]), where an unknown pattern is imposed on the network at time zero, the network iterates in discrete time steps using some given formula that forces the output of the network to match one of the stored patterns. Unlike the feedforward network, output of a feedback network now is considered as a state vector of the network which no longer changes on successive iterations, or a converged state of the network.

- **Optimization problem solver** It was the papers [151], [280] by Hopfield and Tank that initialized applications of modern artificial neural networks on optimization problems: combinatorial optimization and mathematical programming problems. As we will show in this book as a main topic, if a feedback neural network is described by a set of discrete or continuous dynamic systems:

$$y(t+1) = f(y(t)) \quad \text{or} \quad \dot{y} = h(y)$$

(5.18)

with a corresponding Liapunov function $E(y(t))$, then moving along the trajectory of the dynamic system as $t$ increasing will decrease the value of the function $E(y(t))$. For a given optimization problem, if a neural network can be constructed such that (1) it has a corresponding Liapunov function which is equivalent to the objective function or merit function of the optimization problem; (2) the trajectory from a properly chosen starting point will converge to a solution of the optimization problem, then we establish an artificial neural network model for solving this optimization problem or a class of problems to which the given problem belongs. We will discuss this philosophy in details in the main chapters of this book.
学霸图书馆

www.xuebalib.com

本文献由“学霸图书馆-文献云下载”收集自网络，仅供学习交流使用。

学霸图书馆（www.xuebalib.com）是一个“整合众多图书馆数据库资源，提供一站式文献检索和下载服务”的24小时在线不限IP图书馆。

图书馆致力于便利、促进学习与科研，提供最强文献下载服务。

图书馆导航：

图书馆首页 文献云下载 图书馆入口 外文数据库大全 疑难文献辅助工具