3. Neural Network-based Learning and Control

3.1 Introduction

The main focus of this chapter is to describe a number of neural network-based concepts for control problems. The ability of neural networks to learn about their environment and to adaptively fine-tune their parameters to improve the systems' performance is one of their strong points. Being able to model systems or phenomenons allows neural networks to be used in a variety of ways. In control systems applications, feedforward multi-layer neural networks with supervised training are the most commonly used. According to the classical theorem due to Stone-Weierstrass (Girosi and Poggio, 1989), neural networks are able to generate input-output maps that can approximate any function with a desired accuracy. In this view, system identification and control are the two most commonly encountered control problems which neural networks have been utilized to solve. Neural networks can be used as the main controller to solve a given control problem or as a complementary controller for other controllers (e.g., PID controllers).

Some of the common variants (Miller et al., 1990) of neural network-based control are:

- **System identification.** With all advanced control schemes, mathematical knowledge of the dynamics of the plant is necessary. Using experimental data or other empirical information of the plant, a mathematical model of the plant can be obtained via the process of system identification. System identification can be performed in two different ways, i.e., with and without a priori knowledge about the plant. With a priori knowledge about the plant, parametric identification with neural network is performed; whereas, without a priori knowledge about the plant, system identification using the neural network which function as a black box model is carried out.

- **Identification of system inverse.** For this purpose, the neural network is trained as an inverse model of the plant, using supervised learning. The network input is the output of the plant and the network output is the plant input.
• Control autotuning using neural networks. The performance of closed-loop systems is dependent upon the proper tuning of the controller's parameters. For instance, the PID controller is one that is commonly used as a closed-loop controller. Neural networks can be utilized in the autotuning process of these controllers.

In this chapter, the main learning methodologies in neural networks will first be highlighted. With these learning methodologies in mind, the different learning algorithms or rules which use the main learning methodologies as a blueprint will then be described. A section is added in illustrate the novel combination of neural network and fuzzy logic for control applications. Following all these and the neural network fundamentals in Chapter 1, neural networks for control purposes can now be discussed. Identification using neural networks will be discussed next, followed by the different control structures which incorporate neural networks in their configurations. Lastly, an application of Taguchi method to tune the weights of a RBF network is presented, with simulation and experimental results.

3.2 Learning Methodologies

Learning in a neural network is performed to mimic the behaviour of its biological counterpart and is still undergoing intense research. Through learning, a neural network is able to adapt itself and subsequently improve its performance in a gradual manner. The learning process is completed when the neural network is able to produce the desired responses when different inputs (or stimuli) are applied to it. More specifically, the neural network learns or adapts itself by adjusting its parameters (namely, the weights and bias). In some cases, the learning process is online or adaptive in nature. That is to say, the neural network learning process is never-ending and continues with changes in the environment. Some of the important learning methodologies will first be described. With these methodologies as the backbone of discussion, a diverse collection of learning rules/algorithms will be mentioned. Subsequently, some considerations related to neural network learning will be highlighted.

3.2.1 Supervised Learning

Supervised learning is also commonly known as 'Learning with a teacher'. Figure 3.1 shows the structure of this form of learning. The teacher has the knowledge of the environment. A training set, comprising an input vector \( \mathbf{x} \)
and the corresponding desired output vector \( \mathbf{d} \), is presented to the network. With \textit{a priori} knowledge of the environment, the teacher is able to provide the neural network with desired responses for the set of training vectors. The teacher is often an uncontrollable or unknown part of the learning process. The aim in supervised learning is to make the neural network emulate the teacher. The output \( \mathbf{y} \) is produced by the network, and compared with the desired output presented by the teacher. The error \( (\mathbf{e} = \mathbf{d} - \mathbf{y}) \) is then used to adjust the parameters of the network, so as to make the neural network emulate the teacher more and more closely. Usually, many training cycles or epochs (from a few to hundreds of thousands) are required to properly train the network. This type of learning is sometimes known as error-correction learning with the objective of minimizing a mean-square-error cost function or a sum of squared errors over the whole training set. The mean-square-error cost function is

\[
J = E\left[ \frac{1}{2} \sum_{i=1}^{K} e_i^2 \right],
\]

where \( E \) is the statistical expectation operator. The summation is over all the \( K \) neurons in the output layer. When the neural network is able to emulate the teacher well enough, the learning process is then completed. The next step is thus to design the learning algorithm or a set of learning rules (that follows the guidelines of supervised learning) to train the neural network, with an adequate set of input-output data.

![Fig. 3.1. Structure of supervised learning.](image-url)
3.2.2 Unsupervised Learning

In contrast to supervised learning, unsupervised learning (commonly known as ‘Learning without a teacher’) does not require a teacher; that is, there is no desired or target output. Figure 3.2 shows the structure of this form of learning. Input is presented and output computed by network. There is no desired response. Output is used to modify the parameters of the neural network. In unsupervised learning, there is no teacher to oversee the learning process. That is to say, labeled examples of the function to be learned are not available. One common application of this type of learning is to cluster or classify groups of objects.

During the training process, the neural network groups the training patterns into classes. When the neural network is presented with a training pattern, it searches the existing classes and tries to put this training pattern into one of the class. If none of the existing classes can be used to group this new training pattern, a new class will be created for this pattern. Within the same class, all the patterns have similar characteristics or properties (e.g., colour, size, texture and etc). In unsupervised learning, the neural network needs certain selection rules or guidelines to determine how new groups can be formed.

![Fig. 3.2. Structure of unsupervised learning.](image)

3.2.3 Reinforcement Learning

Reinforcement learning mimics biological learning in biological neural networks more closely, as compared to supervised and unsupervised learning. Figure 3.3 shows the structure of this form of learning. As stated in Thorndike’s Law,

“If an action taken by a learning system (animal) is followed by a satisfactory state of affairs, then the tendency of the system (animal) to produce that particular action is strengthened or reinforced. Otherwise, the tendency of the system (animal) to produce that action is weakened”.

![Fig. 3.3. Structure of reinforcement learning.](image)
In simpler terms, the idea in reinforcement learning is to reward when the correct or anticipated behaviour is displayed. Reinforcement learning is an on-line (adaptive) learning process, through trial-and-error to maximize a scalar performance index, i.e., a reinforcement signal. During the training process, the neural network is presented with an input pattern. The teacher does not provide a desired response, with respect to this input pattern. Instead, the teacher provides only a reinforcement signal which is either 'Pass' or 'Fail'. If a 'Fail' signal is received, the neural network's parameters will be adjusted. The process is then repeated until a 'Pass' signal is obtained. Throughout the learning process, there is no indication if the neural network is learning in the correct direction or if the network parameters are near their optimum values. Thus, some kind of stopping criteria should be imposed in this type of learning to prevent the situation of being trapped in an endless loop or unbounded parameter values.

Fig. 3.3. Structure of reinforcement learning.

3.2.4 Competitive Learning (Winner-Takes-All Learning)

Competitive learning is a type of self-organizing learning which differs substantially from the other types of learning. The output layer of the neural network which is trained using competitive learning consists of more than one node. The output nodes compete among themselves to produce the closest output signal to the desired vector when a training vector is applied to the neural network. The winning node (or the dominant node) is the one which has the output closest to the desired vector. Only the winning node's
where $\Delta E$ is the change in $E$ resulting from such a flip.

- If the above steps are repeated, the machine will eventually reach thermal equilibrium.

The Boltzmann machine has two types of neurons, i.e., visible and hidden. The visible neurons interacts with the environment, and thus may be constrained in their states, whereas the hidden neurons operate freely. One of the uses of Boltzmann machine is for pattern completion, i.e., a partial pattern is presented and the machine completes the picture after reaching thermal equilibrium.

### 3.4 Neuro-fuzzy Control

Although neural networks and fuzzy systems are structurally very different, they share a rather complementary nature as far as strengths and weaknesses are concerned. Both of them permits a certain tolerance for imprecision and uncertainty. To integrate these two approaches together, fuzzy logic operations can be incorporated into neural networks (i.e., fuzzification of neural networks) and capabilities of neural networks can be infused into fuzzy systems (i.e., fuzzy systems with online learning adaptation). The result is called a neuro-fuzzy network (or sometimes called a fuzzy neural system or fuzzy artificial neural network). In this section, the name fuzzy neural network will be adopted.

#### 3.4.1 Fuzzy Neural Network

In the fuzzy neural network (Lee and Lee, 1974; 1975), the learning, classification, function approximation are performed by the neural network part whereas the fuzzy logic part performs inference and provides fuzzified outputs. During the training (i.e., learning) phase of this network, the membership functions and fuzzy logic rules are modified and updated.

Different approaches for fuzzification of a neuron have been proposed in the literature. A fuzzy neuron (Figure 3.12) has the same basic structure as the McCulloch-Pitts neuron. However, there are some fundamental differences. In place of scalar weights $w_{ij}$, the fuzzy neuron uses fuzzy sets, and in place of the activation function, a fuzzy set may also be used. In Figure 3.12, the external input vector $x = [x_1, x_2, ..., x_R]^T \in R^R$ is defined over the unit hypercube $[0, 1]^R$ and is comprised of fuzzy signals bounded by graded membership over the unit interval $[0, 1]$. The external inputs, after being modified
by synaptic weights (i.e., \( w_1, w_2, \ldots, w_R \)) (also defined over the unit interval), become the inputs (i.e., \( s_1, s_2, \ldots, s_R \)) to the central processing unit. This modification may be just a simple vector multiplication

\[
s_r = w_r x_r,
\]

(3.87)

(\text{where } 1 < r < R) or taking the maximum of the input values and weight (i.e., an OR-gate)

\[
s_r = w_r \lor x_r,
\]

(3.88)

or the minimum (\( \land \)) of the input values and weight (i.e., an AND-gate)

The inputs (i.e., \( s_1, s_2, \ldots, s_R \)) are processed by a fuzzy aggregation operator \( G_j \) that selects the minimum (\( \land \)) of the product (or max) modifications; for example,

\[
G_j = \land_{r=1}^R s_r.
\]

(3.89)

Fuzzy aggregation operators such as \( MIN \) and \( MAX \) are often used in fuzzy neurons. Each fuzzy neuron can be seen as a realisation of a fuzzy linguistic value such as \( HIGH \), \( LOW \), and so on. The output of the neuron \( y \) could be associated with membership to some linguistic value; that is, \( y_i ([0, 1]) \) expresses the degree to which the input pattern \( x = [x_1, x_2, \ldots, x_R]^T \) belongs to a given linguistic variable. The outputs of all the fuzzy neurons \( y_i \)'s (\( i = 1, 2, \ldots, N \)) can then be combined to realize a fuzzy rule. There are different models of fuzzy neurons that have been proposed (Kwan and Cai, 1994):

- **Max (OR) Fuzzy Neuron**

  A max fuzzy neuron (also called an OR-fuzzy neuron) selects the maximum (\( \lor \)) of the inputs,

  \[
  G_j = \lor_{r=1}^R s_r
  = \lor_{r=1}^R w_r x_r,
  \]

  (3.90)

  A max fuzzy neuron is an implementation of a logical \( OR \).

- **Min (AND) Fuzzy Neuron**

  A min fuzzy neuron selects the minimum (\( \land \)) of the inputs,

  \[
  G_j = \land_{r=1}^R s_r
  = \land_{r=1}^R w_r x_r.
  \]

  (3.91)
3.4.2 Fuzzy Neural Control

To reap the benefits of fuzzy systems and neural networks for control purposes, many novel approaches have been proposed (Gupta and Rao, 1994; Lin and Lee, 1996; Yen et al., 1995). Two possible structures of fuzzy neural networks are as shown in Figures 3.13 and 3.14. The neural networks are able to generate new fuzzy rules and have learning capabilities.

![Fuzzy neuron](image1)

**Fig. 3.12.** Fuzzy neuron.

![Fuzzy neural network](image2)

**Fig. 3.13.** Fuzzy neural network.
3.5 Factors Affecting Learning

Some of the important parameters that affect the learning capability of neural networks and other considerations pertinent to neural network learning will be highlighted in this section.

3.5.1 Learning Constant, $\eta$

- The value of the learning constant, $\eta$, is extremely important in determining not only how fast the neural network learns, but also whether it will learn properly.

- Too small a value for $\eta$ makes learning too slow while a large value for $\eta$ may cause ‘overshooting’ of the solution. Typical values of $\eta$ which have been cited by authors range from 0.001 to 10 with values between 0.1 to 0.9 being common. The “best” value of $\eta$ depends upon the architecture of the network and also on the function it is required to map. When the error surface varies too rapidly (i.e. has high frequency components), then a small value of $\eta$ will work well. For error surfaces which vary slowly or gradually, large values of $\eta$ may be needed. As the error surface is not known (otherwise, the optimum solution can be identified), a good value of $\eta$ will have to be obtained from experience or experimentation.

- Sometimes an adaptive (or changing) learning constant is used to improve the rate of learning. One such method is the decaying learning constant.