Chapter 1
Artificial Neural Networks

An Artificial Neural Network is an information processing system that has certain performance characteristics in common with biological neural networks. Artificial neural networks have been developed as generalizations of mathematical models of human cognition or neural biology [16]. The basic processing elements of neural networks are called neurons. Neuron performs as summing and nonlinear mapping junctions. In some cases they can be considered as threshold units that fire when their total input exceeds certain bias levels. Neurons usually operate in parallel and are configured in regular architectures. The neurons are generally arranged in parallel to form layers. Strength of the each connection is expressed by a numerical value called a weight, which can be modified [17].

Each neuron in a layer has the same number of inputs, which is equal to the total number of inputs to that layer. Therefore, every layer of p neurons has a total of n inputs and a total of p outputs (each neuron has exactly one output). This implies that each neuron has n inputs as well. In addition, all the neurons in a layer have the same nonlinear transfer function. Each neuron has a number of weights, each corresponding to an input, along with a threshold or bias value. The list of weights can be thought as a vector of weights arranged from 1 to n, the number of inputs for that neuron. Since each neuron in a layer has the same number of inputs, but a different set of weights, the weight vectors from p neurons can be appended to form a weight matrix W of size (n, p).Similarly, the single bias constant from each neuron is appended into a vector of length p. Note that *every* neuron in the net is identical in structure. This fact suggests that the neural net is inherently massively parallel [18].

1.1 Artificial Neural Network Cell (Perceptron)

Artificial neural network consisting of a single layer and a simple activation function is called perceptron. The first formal definition of a synthetic neuron model based on the highly simplified consideration of the biological model was formulated by McCulloch and Pitts [17]. A basic artificial neural network neuron has more simple structure than a biological neuron cell. Biological neuron cell and basic neuron model are shown in Figure 1.1.

The scalar input p is multiplied by the scalar weight w to form w_p, one of the terms that is sent to the summer. The other input, 1, is multiplied by a bias b and then passed to the summer. The summer output, n, often referred to as the net input, goes into a transfer function or activation function f, which produces the scalar neuron, output a [19].

M. Ünal et al.: Optimization of PID Controllers Using AC and GA, SCI 449, pp. 5–17.
springerlink.com

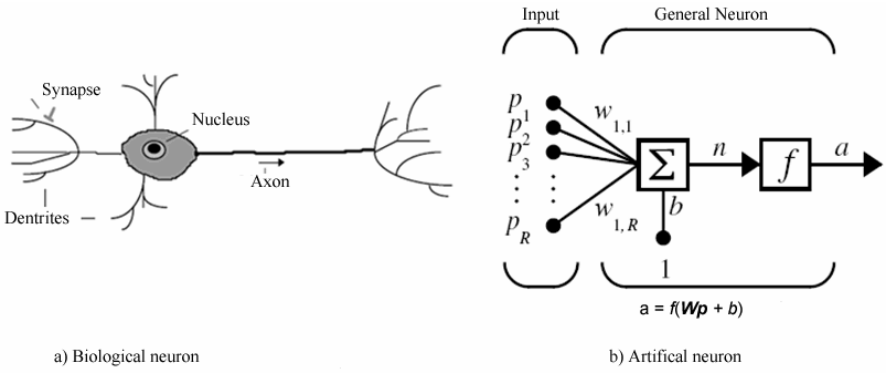


Fig. 1.1 Biological and Artificial Neurons

Typical activation functions used are

$$f(net) \triangleq \frac{2}{1+\exp(-\lambda net)} - 1 \tag{1.1a}$$

and

$$f(net) \triangleq \text{sgn}(net) = \begin{Bmatrix} +1, & net > 0 \\ -1, & net < 0 \end{Bmatrix} \tag{1.1b}$$

where $\lambda > 0$ in (1.1a) is proportional to the neuron gain determining steepness of the continuous function *f(net)* near $net = 0$. The continuous activation function is shown in Figure 1.2a for various. As $\lambda \to \infty$ the limit of the continuous function becomes the *sgn (net)* function. Activation functions Equation 1.1a and 1.1b are called bipolar continuous and bipolar binary functions, respectively. The word "bipolar" is used to point out that both positive and negative responses of neurons are produced for this definition of the activation function.

By shifting and scaling the bipolar activation functions defined by Eq.1.1, unipolar continuous and unipolar binary activation functions can be obtained, respectively, as follows

$$f(net) \triangleq \frac{1}{1+\exp(-\lambda net)} \tag{1.2a}$$

and

$$f(net) \triangleq \text{sgn}(net) = \begin{Bmatrix} 1, & net > 0 \\ 1, & net < 0 \end{Bmatrix} \tag{1.2b}$$

Function 2a is shown in Figure 1.2a. The unipolar binary function is the limit of *f(net)* in Equation 1.2a when $\lambda \to \infty$. The soft-limiting activation functions Equation 1.1a and 1.2a are often called sigmodial characteristics, as opposed to the hard-limiting activation functions given in Equation 1.1b and 1.2b [17].

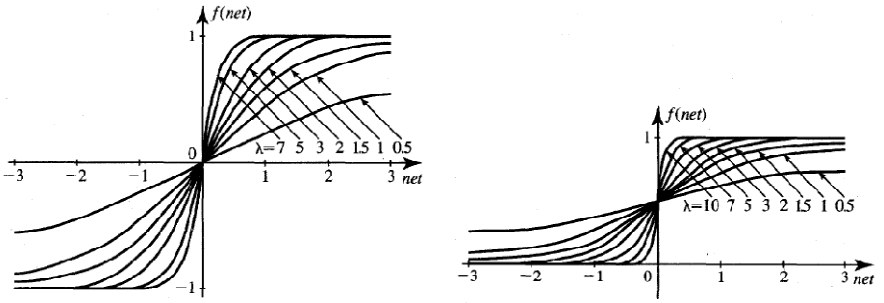


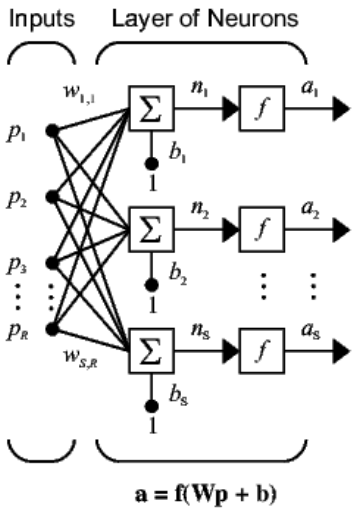
Fig. 1.2 Activation functions of a neuron (a) bipolar continuous (b) unipolar continuous

1.2 Artificial Neural Network Models

ANN models can be examined in two groups according to the structure: Feedforward and Recurrent networks.

1.2.1 Feedforward Network Models

The simplest and most common artificial neural networks use one-way signal flow. There are no delays in the feed forward artificial neural networks, the process advances to the inputs to outputs. Output values are compared with the desired output value and weights are updated with obtained error signal. Feedforward network model is shown in Figure 1.3.



Where

R = number of elements in vector

S = number of neurons in layer

Fig. 1.3 Layer of S neurons

In this network, each element of the input vector p is connected to each neuron input through the weight matrix W. The ith neuron has a summer that gathers its weighted inputs and bias to form its own scalar output $n(i)$. The various $n(i)$ taken together form an S-element net input vector n. Finally, the neuron layer outputs form a column vector a. The expression for a is shown at the bottom of the figure.

Note that it is common for the number of inputs to a layer to be different from the number of neurons (i.e., R is not necessarily equal to S). A layer is not constrained to have the number of its inputs equal to the number of its neurons.

The input vector elements enter the network through the weight matrix W.

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{1,1} \\ \cdot & \cdot & & \cdot \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} \tag{1.3}$$

Note that the row indices on the elements of matrix W indicate the destination neuron of the weight, and the column indices indicate which source is the input for that weight. Thus, the indices in $w_{1,2}$ say that the strength of the signal from the second input element to the first (and only) neuron is $w_{1,2}$.

A network can have several layers. Each layer has a weight matrix W, a bias vector b, and an output vector a. To distinguish between the weight matrices, output vectors, etc., for each of these layers in the figures, the number of the layer is appended as a superscript to the variable of interest. A three-layer network is shown in Figure 1.4.

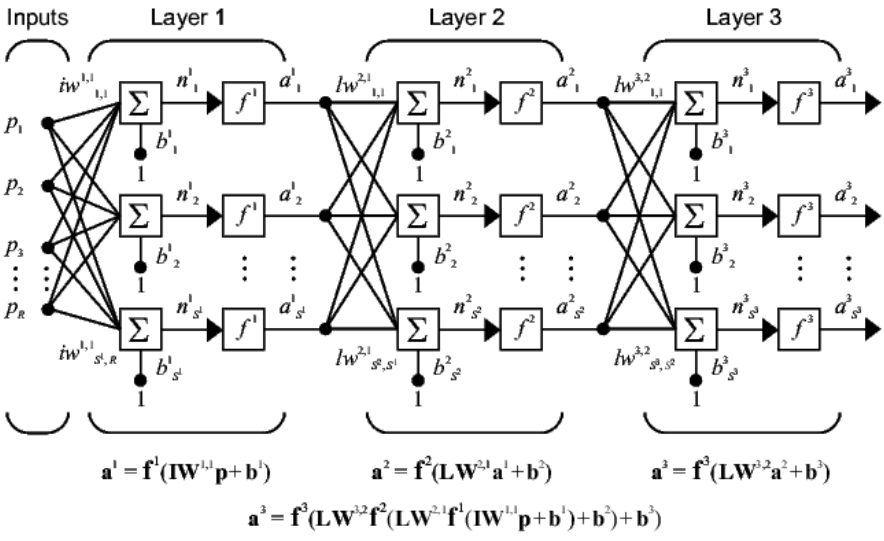


Fig. 1.4 Three layer network

The network shown above has R inputs, S^1 neurons in the first layer, S^2 neurons in the second layer, etc. It is common for different layers to have different numbers of neurons. A constant input 1 is fed to the bias for each neuron.

Note that the outputs of each intermediate layer are the inputs to the following layer. Thus layer 2 can be analyzed as a one-layer network with $R = S^1$ inputs, $S = S^2$ neurons, and a $S^1 x S^2$ weight matrix $\boldsymbol{W}^2$. The input to layer 2 is a^1; the output is a^2. Now, all the vectors and matrices of layer 2 have been identified, it can be treated as a single-layer network on its own. This approach can be taken with any layer of the network.

The layers of a multilayer network play different roles. A layer that produces the network output is called an output layer. All other layers are called hidden layers.

1.2.2 Recurrent Networks

A feedback network can be obtained from the feedforward network by connecting the neurons' outputs to their inputs. There are delays in artificial neural networks with feedback. One type of discrete-time recurrent network is shown in Figure 1.5.

D shows the time delay. D holds the information at the past sample time in memory and it is applied to the next output as input.

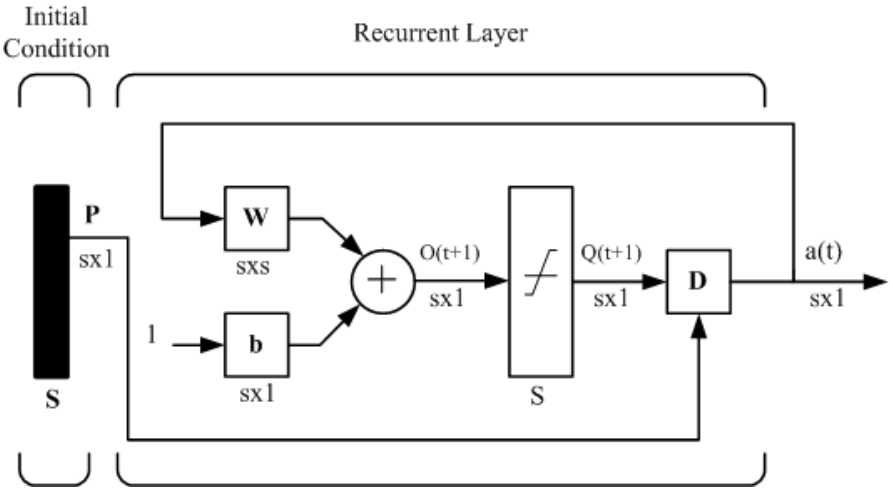


Fig. 1.5 Recurrent Network

1.2.3 Learning Rules

A learning rule is defined as a procedure for modifying the weights and biases of a network (This procedure can also be referred to as a training algorithm). The learning rule is applied to train the network to perform some particular tasks. Learning rules fall into three broad categories: supervised learning, unsupervised

learning and reinforcement (or graded) learning. The block diagrams of supervised learning and unsupervised learning are illustrated in Figure 1.6 [17].

In supervised learning we assume that at each instant of time when the input is applied, the desired response d of the system is provided by the teacher. This is illustrated in Figure 6(a). The distance *p[d, o]* between the actual and the desired response serves as an error measure and is used to correct network parameters externally. Since we assume adjustable weights, the teacher may implement a reward-and-punishment scheme to adapt the network's weight matrix W. This mode of learning is very pervasive. Also, it is used in many situations of natural learning. A set of input and output patterns called a training set is required for this learning mode. Typically, supervised learning rewards accurate classifications or associations and punishes those which yield inaccurate responses. The teacher estimates the negative error gradient direction and reduces the error accordingly. In many situations, the inputs, outputs and the computed gradient are deterministic, however, the minimization of error proceeds over all its random realizations. As a result, most supervised learning algorithms reduce to stochastic minimization of error in multi-dimensional weight space [17].

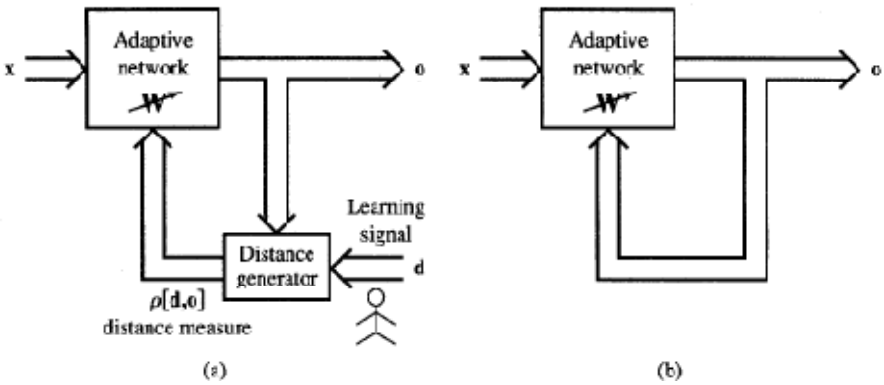


Fig. 1.6 Block diagram for explanation of basic learning modes: (a) Supervised learning and (b) Unsupervised learning

Figure 6(b) shows the block diagram of unsupervised learning. In learning without supervision, the desired response is not known; thus, explicit error information cannot be used to improve network behavior. Since no information is available as to correctness or incorrectness of responses, learning must somehow be accomplished based on observations of responses to inputs that we have marginal or no knowledge about. Unsupervised learning algorithms use patterns that are typically redundant raw data having no labels regarding their class membership, or associations. In this mode of learning, the network must discover for itself any possibly existing patterns, regularities, separating properties, etc. While discovering these, the network undergoes change of its parameters, which is

called self-organization. The technique of unsupervised learning is often used to perform clustering as the unsupervised classification of objects without providing information about the actual classes. This kind of learning corresponds to minimal a priori information available. Some information about the number of clusters, or similarity versus dissimilarity of patterns, can be helpful for this mode of learning.

Unsupervised learning is sometimes called learning without a teacher. This terminology is not the most appropriate, because learning without a teacher is not possible at all. Although, the teacher does not have to be involved in every training step, he has to set goals even in an unsupervised learning mode [20]. It may think of the following analogy. Learning with supervision corresponds to classroom learning with the teacher's questions answered by students and corrected, if needed, by the teacher. Learning without supervision corresponds to learning the subject from a videotape lecture covering the material but not including any other teacher's involvement. The teacher lectures directions and methods, but is not available. Therefore, the student cannot get explanations of unclear questions, check answers and become fully informed [17].

Reinforcement learning is a learning method for agents to acquire the optimal policy autonomously from the evaluation of their behavior. The indication of the performance is called the reinforcement signal. In this learning scheme, the learning agent receives sensory input from the environment. Then it decides and carries out an action based on the input. In return, it receives a reinforcement signal, and new sensory input. The agent uses these episodes to maximize the reinforcement signals aggregated in time [21].

1.2.4 Multilayer Perceptron

Multilayer Perceptron (MLP) consists of three-layers of neurons (input, hidden and output layer as shown in Figure 1.7) interconnected by weights. Neural networks are those with at least two layers of neurons-a hidden and an output layer, provided that the hidden layer neurons have nonlinear and differentiable activation functions. The nonlinear activation functions in a hidden layer enable a neural network to be a universal approximator. Thus, the nonlinearity of the activation functions solves the problem of representation. The differentiability of the hidden layer neurons activation functions solves the nonlinear learning task. Generally, no processing will occur in the input layer.

The input layer is not treated as a layer of neural processing units. The input units are merely fan-out nodes. Generally no processing will occur in the input layer, and although in it looks like a layer, it’s not a layer of neurons. Rather, it is an input vector, eventually augmented with a bias term, whose components will be fed to the next (hidden or output) layer of neural processing units. The output layer neurons may be linear or they can be having sigmoidal activation functions [22].

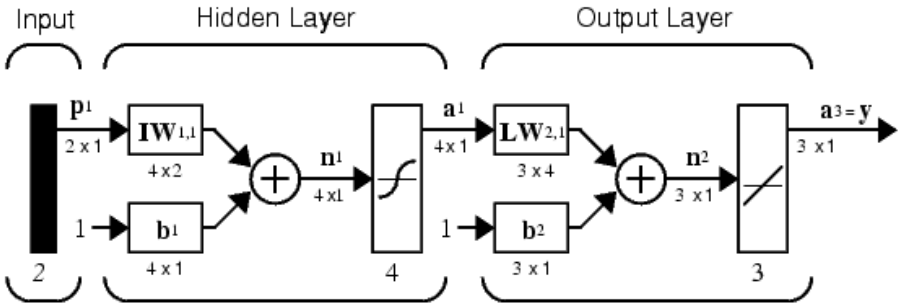


Fig. 1.7 Multi Layer Perceptron

1.2.5 Error Backpropagation Algorithm

The basic idea behind the error backpropagation algorithm is that the error signal terms for hidden layer neurons are calculated by backpropagation the error signal terms of the output layer neurons. Backpropagation is still the most commonly used learning algorithm in the field of soft computing.

MLP network starts with random initial values for its weights, and then computes a one-pass backpropagation algorithm at each time step k, which consists of a forward pass propagating the input vector through the network layer by layer, and a backward pass to update the weights by the gradient descent rule. After having trained on-line for a period of time, the training error should have converged to a value so small that if training were to stop, and the weights frozen. Also, the training of NNs is said to have reached a global minimum when, after changing the operating conditions, as well as freezing the weights, the network's response is still reasonably acceptable [20]. Steps of the algorithm are as following:

Initialize connection weights into small random values.

Present the *p*th sample input vector of pattern $X_p = (X_{p1}, X_{p2}, ..., X_{pN_0})$ and the corresponding output target $T_p = (T_{p1}, T_{p2}, ..., T_{pN_M})$ to the network.

Pass the input values to the first layer, layer 1. For every input node *i* in layer 0, perform:

$$Y_{0i} = X_{pi} \tag{1.4}$$

For every neuron *i* in every layer $j = 1,2,...M$, from input to output layer, find the output from the neuron:

$$Y_{ji} = f\left(\sum_{k=1}^{N_{j=1}} Y_{(j-1)k} W_{jik}\right) \tag{1.5}$$

where *f* is the activation function. The most widely used activation function for neurons in the hidden layer is the following sigmoidal function:

$$f(x) = \frac{1}{1+e^{-x}} \tag{1.6}$$

Obtain output values. For every output node *i* in layer *M*, perform:

$$O_{pi} = Y_{Mi} \tag{1.7}$$

Calculate error value δ_{ji} for every neuron i in every layer in backward order $j = M, M-1, \ldots 2,1$ from output to input layer, followed by weight adjustments. For the output layer, the error value is:

$$\delta_{Mi} = Y_{Mi}(1-Y_{Mi})(T_{pi} - Y_{Mi}) \tag{1.8}$$

and for hidden layers:

$$\delta_{Mi} = Y_{ji}(1-Y_{ji})\sum_{k=1}^{N_{j+1}} \delta_{(j+1)k} W_{(j+1)ki} \tag{1.9}$$

The weight adjustment can be done for every connection from neuron *k* in layer *(i-1)* to every neuron *i* in every layer *i*:

$$W_{jik}^{+} = W_{jik+} \beta \delta_{ji} Y_{ji} \tag{1.10}$$

where β represents weight adjustment factor normalized between 0 and 1.

The actions in steps 2 through 6 will be repeated for every training sample pattern *p*, and repeated for these sets until the Root Mean Square (RMS) of output errors is minimized.

RMS of the errors in the output layer defined as:

$$E_p = \frac{1}{2}\sum_{j=1}^{N_m} \left(T_{pj} - O_{pj}\right)^2 \tag{1.11}$$

for the *p*th sample pattern [21].

1.2.6 The Nonlinear Autoregressive Network with Exogenous Inputs (NARX) Type ANN

A simple way to introduce dynamics into network consists of using an input vector composed of past values of the system inputs and outputs. This the way by which the MLP can be interpreted as the nonlinear autoregressive network with exogenous inputs (NARX) model of the system. This way of introducing dynamics into a static network has the advantage of being simple to implement [23].

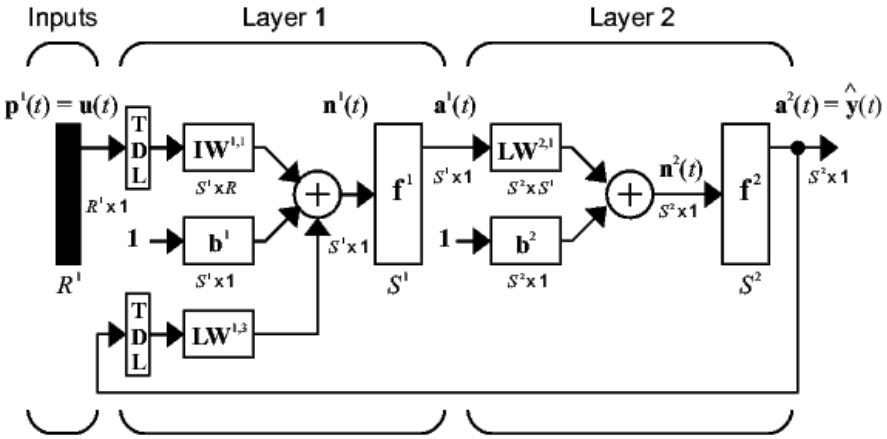


Fig. 1.8 NARX model

NARX model can be implemented by using a feedforward neural network to approximate the function f. A diagram of the resulting network is shown below, where a two-layer feedforward network is used for the approximation.

The defining equation for the NARX model is

$$\hat{y}(k) = f(y(k-1),..., y(k-n); u(k-1),..., u(k-m)) + \varepsilon(k) \tag{1.12}$$

where $\hat{y}(k)$ is predicted value, $y(k)$ process output, $u(k)$ process input, $\varepsilon(k)$ is the approximation error at time instant k, m is an input time delay, n is an output time delay, f is a non-linear function (activation function) describing the system behavior.

There are many applications for the NARX network. It can be used as a predictor, to predict the next value of the input signal. It can also be used for nonlinear filtering, in which the target output is a noise-free version of the input signal. The use of the NARX network is demonstrated in another important application, the modeling of nonlinear dynamic systems.

It can be considered the output of the NARX network to be an estimate of the output of some nonlinear dynamic system. The output is fed back to the input of the feedforward neural network as part of the standard NARX architecture, as shown in the left Figure 1.9. Because the true output is available during the training of the network, it can be created a series-parallel architecture, in which the true output is used instead of feeding back the estimated output, as shown in the right Figure 1.9. This has two advantages. The first is that the input to the feedforward network is more accurate. The second is that the resulting network has a purely feedforward architecture, and static backpropagation can be used for training.

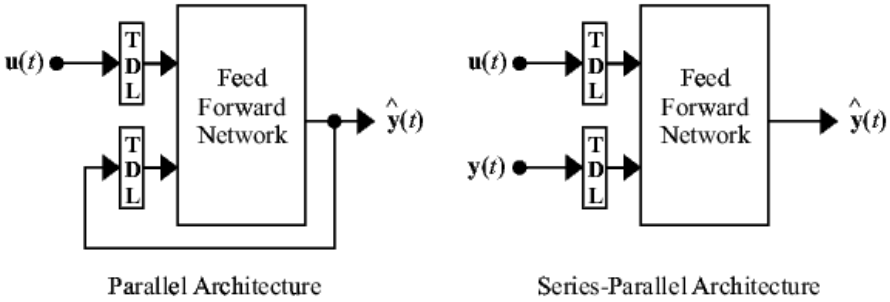


Fig. 1.9 A series-parallel architecture [19]

1.3 Modeling of a System with ANN

Modeling and measurement has very great significance on systems design and control. All elements and link styles of systems must have analytical expressions based on physical laws in classic modeling concept. In fact, this process is not possible or very difficult in complex system. So, in the modern concept of modeling, an appropriate model is proposed to the system and this model is taught with the help of input-output data set and updated. In recent years, especially in complex and nonlinear systems modeling, ANN has been used widely. ANN's nonlinear structure provides a good fit with systems to be taught.

In general, obtaining the system model means that learning of the physical structure of the system based measurement of input, output and state variables. One or more state variables often can be system output. A system model can be specified using the mathematical expression, input-output data set or linguistic rules. ANN can be used for modeling linear and nonlinear systems.

1.3.1 Advanced Modeling (Series-Parallel Modeling)

The input and output changing over time of a dynamic system are shown as, u and y_p respectively. If the model and the system are running the same system input u, $(\hat{y})$'s output produced by the model should be closer to y_p. Block diagram of this method is shown in Figure 1.10.

Identification Artificial Neural Network (ANN) blocks in Figure 1.10, is placed in parallel with the non-linear system. The error between system output y_p and ANN output $\hat{y}(k)$, $e(k) = y_p - \hat{y}$, is used in training of ANN. It's assumed that system is described by the following non-linear discrete-time difference equation.

$$y_p(k+1) = F\left[y_p(k), \ldots, y_p(k-n+1); u(k), \ldots, u(k-m+1) \right] \tag{1.13}$$

Here, $y_p(k+1)$ is the value at the $(k+1)$ sampling time of system output, $F(.)$ is multi-dimensional system definition dependent to past system output (n) and past input (m) where n and m are positive integers. It's assumed that ANN 's input-output structure is the same with system in input-output structure selection [24]. So, ANN output's can be indicated as follows:

$$\hat{y}(k+1) = y_p'(k+1) = N\left[y_p(k),\ldots,y_p(k-n+1);u(k),\ldots,u(k-m+1)\right] \tag{1.14}$$

Here, $N(.)$ is nonlinear definition of the ANN and $\hat{y}$ is the predicted ANN output. As can be seen here, ANN output depends on the past n system output and m system input.

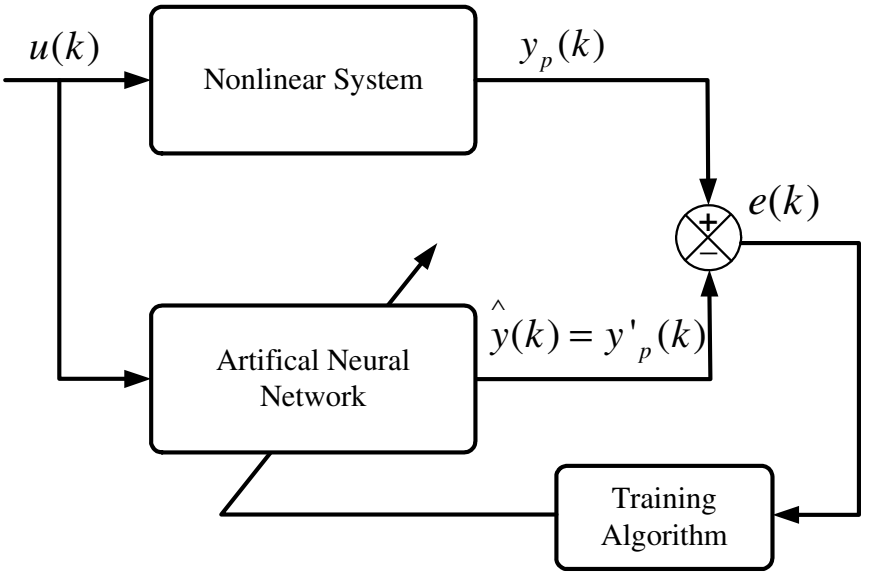


Fig. 1.10 The basic advanced type block diagram using on modeling with ANN

Figure 1.11 shows the block diagram of the identification algorithm created by using Equation 1.13 and 1.14. In this serial-parallel identification system, the main principle is to provide the system output and its backward modes as input to ANN model. Time Delay Layer (TDL) blocks in Figure 1.11 refer cycles of time delay [24-26].

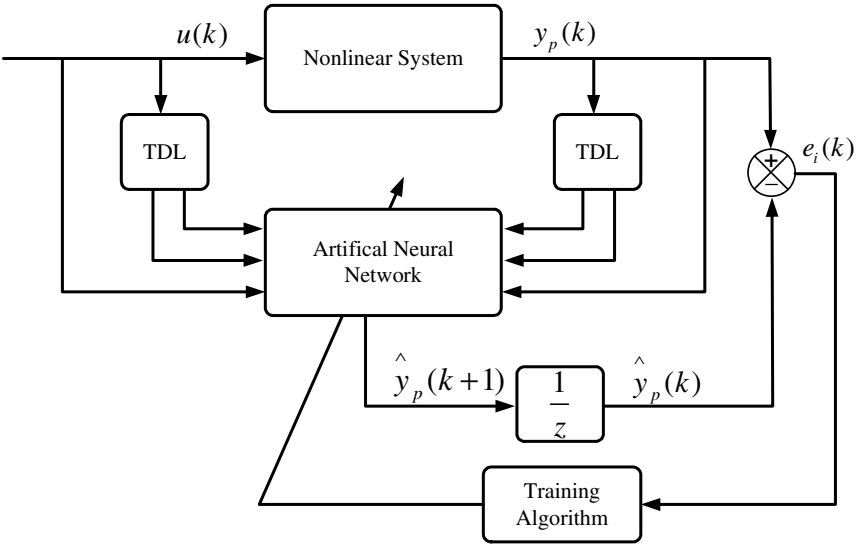


Fig. 1.11 Block diagram of serial – parallel advance type modeling