CHAPTER 2

PROCESS CHARACTERISTICS AND CONTROL STRATEGIES

In this chapter, the process characteristics and the methods to obtain them using simple process information are presented. The different types of control strategies using adaptive control, fuzzy logic and neural network architectures are also presented. These are very useful in tuning controllers, evaluating control system stability and in designing advanced control strategies.

2.1 TYPES OF PROCESS

A process can be classified into self-regulating and non-self-regulating depending on how they respond to an input change. The self-regulating process has a typical response to step change as shown in Figure 2.1. The process undergoes a finite change in output in response to a bounded change in input. The output reaches a new operating point and remains there. That is, the process regulates itself to a new operating condition.
A non-self-regulating process is one, in which, upon a bounded change in input, the output does not regulate itself to a new operating condition. Two different responses of a non-self-regulating process are presented in Figures 2.2 and 2.3. In the first case, the output has a constant rate of change (slope) and in the second case the output changes exponentially. The process with the response depicted in Figure 2.2 is generally referred to as an integrating process. An example of this process is the liquid level in a tank, as shown in Figure 2.4. As the control signal to the outflow valve is decreased, the level in the tank (process output) increases and attains a steady rate of change. Processes with the response shown in Figure 2.3 are referred to as open loop unstable.
Figure 2.2 Servo response of a non-self-regulating integrating process 
(step response)

Figure 2.3 Step response of a non-self-regulating open-loop unstable 
process
2.2 PROCESS CHARACTERISTICS

The three important characteristics of a process that represent the system behavior are the (i) process gain (ii) process time constant and (iii) dead time.

2.2.1 Process gain (K)

The gain is a steady state characteristic of the process and is defined as the ratio of the change in output to the change in input or the excitation function. The process gain is a measure of the sensitivity of the output variable to a change in the input variable. The process gain can be positive or negative and specifies the direction in which the input and output variables change. The gain does not specify the dynamics of the process, that is, it does not give any information as to how fast the process variations occur.
2.2.2 Process time constant ($\tau$)

This is a measure of how fast the process responds to a change in input. This term relates the dynamics of the process. The speed of response of a process and the time constant $\tau$ are inversely related.

2.2.3 Process dead time ($t_d$)

This is defined as the finite amount of time between the change in input variable and the resultant change in the output variable. The parameters $\tau=1.5(t_2-t_1)$ and $t_d$ are shown in Figure 2.5. The numerical values of $K$, $\tau$ and $t_d$ depend on the physical parameters of the process, such as size and calibration.

![Figure 2.5 Servo response of a plant used in the measurement of $t_d$ and $\tau$](image)
2.3 PROCESS NONLINEARITIES

A linear process is one in which the numerical values of $K$, $\tau$ and $t_d$ are constant over the entire operating range. If one or more of these parameters vary within the operating range, they are classified as nonlinear processes. An example of the nonlinear process is shown in Figure 2.6. The Figure shows a conical tank. Since the cross section of the tank at height $h_1$ is less than at $h_2$, the level at $h_1$ will respond faster (to changes in inlet or outlet flow) than at level $h_2$. Thus, the dynamics of the process are faster at $h_1$ than at $h_2$. In other words, the gain varies as a nonlinear function of the pressure drop across the valve, which in turn depends on the liquid head in the tank. The shape of the tank contributes to the nonlinearity in the above process. The identification of the process characteristics for a nonlinear system is required for designing a parameter adaptive controller, in which the controller tunings will vary with operating conditions.

Figure 2.6 An example of a nonlinear process
2.4 FIRST-ORDER-PLUS-DEAD-TIME (FOPDT) TRANSFER FUNCTION MODEL OF A SELF-REGULATING PROCESS

The transfer function model of a self-regulating process (not integrating process) is generally described in terms of the three process characteristics $K$, $\tau$ and $t_d$. Higher order processes with multiple time constants in cascade, are best approximated by a FOPDT transfer function. Thus higher order processes are commonly approximated by low-order-plus-dead-time model. The steps required to obtain the necessary process data are (i) The controller is set in the manual mode, that is, effectively the controller is removed (this is shown in Figure 2.7) (ii) A step signal is given to the plant and (iii) The process variations are recorded. From the process curve the process characteristics are obtained using the two-point method. This method is also known as the process reaction curve method (PRC). A typical PRC of a higher order self-regulating process has already been shown in Figure 2.5.

Figure 2.7 Block diagram of the control system used to obtain the PRC
From the PRC, the process characteristics $r$, $t_d$ and $K$ are obtained as

(i) $r = 1.5(t_{0.632\delta_0} - t_{0.283\delta_0})$

(ii) $t_d = (t_{0.632\delta_0} - r)$ and

(iii) $K = \frac{\delta_o}{\delta_{co}}$ where $\delta_o$ is the change in the process output and $\delta_{co}$ is the change in the controller output or process input.

Once these three process characteristics are identified, the process transfer function is defined as $\frac{C(s)}{M(s)} = \frac{(Ke^{st_d})}{(sT+1)}$ where $C(s)$ is the process output and $M(s)$ is the controller output (process input).

### 2.5 PI CONTROLLER

The commonly used controllers in industrial processes are the Ziegler-Nichols tuned PI controllers with fixed parameters. The transfer function of the PI controller is given by

$$G_c(s) = K_c \left[1 + \frac{1}{sT_i}\right]$$

The limitations of these controllers are that, they exhibit poor performance when applied to systems containing nonlinearities such as, dead zone, hysteresis and saturation. The usual way to optimize the control action in such situations is to retune the PI settings. However, this cannot cope with time varying system parameters or system nonlinearities.
2.6 ADAPTIVE CONTROL THEORY

Adaptive control is an important area of modern control used in the control of processes with changing parameters. The growth of interest in this field during the past three decades has considerably increased. Adaptive control techniques have great potential, as these methods can cope with increasingly complex systems in the presence of extreme changes in the system parameters and input signals. These controllers have a structure similar to PI controllers, but their parameters are adapted on-line based on parameter estimation requiring certain knowledge of the process.

2.6.1 Motivation for using Adaptive Control

A real-world plant is usually characterized by time-varying dynamical properties, most of them caused by the plant (i) being non-stationary and (ii) non-linear besides random disturbances. It is clear that the control algorithm used in these circumstances should either be adaptive or exhibit the required robustness with respect to poor plant models and changes in the plant dynamics. Robustness can generally be ensured by the feedback structure of the control system. The feedback compensates for the deviation of the plant output from its set point, irrespective of the factors that has caused deviation. The common causes for deviations are (i) exogenous disturbances entering the plant (ii) improper plant model structure and (iii) perturbations in the parameters of the plant model. However, the two latter factors cannot always be dealt with well enough by the feedback structure alone. Wide differences in plant model structure, along with variations in dynamics, can impair the robustness of the system and thereby degrade its performance. These structural perturbations and
environmental variations that result in the degradation of system performance can be cited in a number of examples including the process control where there are changes in the rates of inflow and/or outflow. In brief, the use of adaptive control is very much worth a trial in systems with (i) modifications in the plant transfer function, either in its order or in the value of some parameters due to variations in the environment, the size and properties of raw materials, the plant throughput, the characteristics involving alterations in the coefficients and wear and tear of some important components, (ii) stochastic disturbances and variations in the nature of inputs, (iii) nonlinear behaviour as in the case of complex chemical or bio-chemical reactions, (iv) systems having appreciable dead time and (v) unknown parameters existing when a control system for a new process is commissioned.

2.6.2 Essential aspects of Adaptive Control

The basic functions common to most adaptive control systems are the following:

(i) To identify the unknown parameters of a plant or to measure an index of performance (IP)
(ii) To take appropriate decision on the control strategy
(iii) To perform on-line modification of the parameters of the controller or the input signal.

Depending on how these functions are brought about, different types of adaptive controllers are categorized. To outline the essential aspects of adaptive control, the following definition is considered. "An adaptive system
measures a certain index of performance (IP) using the inputs, the states and the outputs of the adjustable system. From the comparison of the measured IP values and a set of given ones, the adaptation mechanism modifies the parameters of the adjustable system or generates an auxiliary input in order to maintain the IP values close to the set of given ones.

The usual techniques of changing the controller parameters are through gain scheduling and plant model identification.

2.6.3 Classification of Adaptive Control

Adaptive control systems may be classified according to the

(i) adaptation mechanism: parameter-adaptive or signal-synthesis adaptive,

(ii) operating conditions: deterministic, stochastic or learning systems,

(iii) nature of the basic mathematical equations used: linear, distributed, parameter, continuous-time, discrete-time systems or hybrid systems,

(iv) index of performance specified: static, dynamic, parametric or function of state variables and inputs,

(v) choice of test signals used to measure the index of performance correctly and quickly,

(vi) nature of comparison-decision block,
(vii) **nature of uncertainty**: parametric or structural uncertainty and time-invariant or time-varying uncertainties and

(viii) nature of constraints imposed either on the desired system or on the adaptation process employed.

There are two principal approaches in designing adaptive controllers, namely, **Model Reference Adaptive Control (MRAC)** and **Self-Tuning Controllers (STC)**. MRAC can be classified further as parameter adaptation type and signal synthesis type.

In the MRAC parameter adaptation type (Figure 2.8), the error due to the deviation between the output of a reference model and an adjustable system (plant) is used by the adaptation mechanism to modify the controller parameters with an objective to minimize the error. Instead, if the adaptation mechanism is used to generate an auxiliary input signal with an objective to

![Figure 2.8 Model Reference Adaptive Controller (Parameter-Adaptation Type)]
minimize the difference between the output of the reference model and that of the adjustable system then the system (plant) represents a signal-synthesis MRAC (Figure 2.9). The main problem here is to determine the adaptation mechanism, so as to achieve a negligible error together with stability. If stability is not ensured, the adaptive system is not of any practical utility.

2.6.4 Self-Tuning Controllers (STC)

Another important form of adaptive control system (Astrom et al 1995) is the STC. In an STC, a design procedure for known plant parameters is first chosen. This is applied to the unknown plant using recursively-estimated values of the parameter. Such a system is shown in Figure 2.10. The regulator design (see Figure 2.10)
represents the on-line solution for a system with known parameters. This is called the "underlying design problem". To evaluate adaptive control schemes, it is helpful to find this underlying design problem, as this will give the characteristics of the system under ideal conditions where the parameters are known exactly. The perturbation signals are not shown in Figure 2.10. However, these are necessary to obtain good estimates. STC's have become very popular in recent years because of the ease with which they can be implemented with microprocessors. The STC shown in figure is called an explicit STC. It is based on the estimation of an explicit process model. Sometimes, the process may be represented in terms of the regulator parameters. This results in simpler algorithms as the design calculations are avoided. Such an STC is called an implicit STC, since it is based on estimation of an implicit process model.
MRAC can be used with continuous-time as well as discrete-time systems, whereas STC involves only discrete-time systems. MRAC systems are mostly applied in deterministic systems whereas, there has been a considerable progress in the development of stochastic STC's.

2.7 FUZZY LOGIC CONTROL

Fuzzy logic has emerged as one of the most active and fruitful areas for research in real time industrial processes, which do not lend themselves to control by conventional methods because of a lack of quantitative data relating the input and output. Fuzzy control is based on fuzzy logic, a logic which is much closer in spirit to human thinking and natural language unlike traditional logic systems.

2.7.1 Fuzzy Knowledge Based Controllers (FKBC)

The Fuzzy Knowledge-Based Controller is shown in Figure 2.11. It includes four major components namely

(i) Knowledge-Base
(ii) Fuzzification
(iii) Inference Mechanism and
(iv) Defuzzification.
2.7.2 Fuzzification

Fuzzification converts a crisp input signal into fuzzified signals that can be identified by grade of membership in fuzzy sets. It involves

a) measuring the values of plant variables,

b) performing a scale mapping that transfers the range of values of the measured input variables into corresponding universe of discourse and

c) assigning suitable linguistic values for the input data, which may be viewed as labels of fuzzy sets.
2.7.3 **Knowledge-Base and Rule-Base**

The database is composed of a knowledge-base and rule-base. The knowledge-base, consisting of input and output membership functions, along with the rule-base provides information for the appropriate fuzzification operations, the inference mechanism and defuzzification. The rule-base consists of a set of linguistic control rules relating the fuzzy input variables to the desired fuzzy control actions that are mostly derived based on

1. operator experience and control engineering knowledge
2. operations and control actions
3. fuzzy model of a process
4. learning.

These four approaches are not mutually exclusive and one or more of their combinations may be needed to construct an effective method for the derivation of fuzzy rules. The basic function of a rule-base is to represent a set of prediction rules in a structured way such as

**IF (process state) THEN (control output)**

The design parameters involved in the construction of the rule-base includes the following:

1. Choice of process state and control output variables
2. Choice of the contents of the rule antecedent and the rule consequent
3. Choice of term sets (range of linguistic values) for the process state and control output variable and

4. Derivation of the set of rules.

2.7.4 Inference Mechanism

An Inference mechanism is the kernel of fuzzy logic controller. The control rules are evaluated by an inference mechanism. There are two basic approaches employed in the design of the inference engine of Fuzzy Knowledge Based Controller. (1) Composition based inference (firing) and (2) Individual rule based inference (firing). Of the two, the second one is mostly used. The basic function of the inference engine (2) is to compute the overall values of the control output variable depending on the individual contribution of each rule fired in the rule base. There are several methods, such as the min-max algorithm, the correlation product algorithm, the Mamdani algorithm etc., for the implementation of the inference mechanism.

2.7.5 Defuzzification

Defuzzification is a technique of converting the final combined fuzzy conclusion into a crisp one. The defuzzified output is in turn applied to the plant.

The following four strategies are commonly used in defuzzification:

1. Max criterion method
2. Mean of maximum method
3. Center of area method (center of gravity) and
4. Weighted average method.
Of the above four methods, the center of area method (center of gravity) is the most widely used technique since the defuzzified value tends to move smoothly around the output fuzzy region, i.e., changes in fuzzy set topology from one model to the next usually results in smooth transitions in the expected value.

2.7.6 Advantages of Fuzzy Logic Controller

Fuzzy logic is best applied to nonlinear, time-variant and ill-defined systems. In system control, the fuzzy approach has a distinct edge over conventional methods. "Preprocessing" large values into a small number of membership grades reduces the number of values that a controller has to contend with to make a decision. Fuzzy logic deals with observed variables rather than the measured variables of system. This means that more variables can be indirectly evaluated than a conventional PID controller. Implementing a control system with fuzzy logic can reduce design complexity to a point where previously insolvable problems can now be solved. Fuzzy systems typically result in a 10:1 rule reduction, requiring less software to implement the same decision-making capability.

2.8 NEURAL NETWORKS AND THEIR CONTROL STRUCTURES

Here, the possibilities of using neural networks for nonlinear control are reviewed. The neural networks are generally viewed as process modelling formalism or even a knowledge representation framework. The knowledge about the plant dynamics and mapping characteristics are implicitly stored
within the network. The nonlinear functional mapping properties of neural networks are central to their use in control applications. Training a neural network using input-output data from a nonlinear plant is considered as a nonlinear functional approximation problem.

### 2.8.1 Basic Neural Learning Model

One type of neural structure used for learning and control is shown in Figure 2.12. To cope with uncertainties regarding plant dynamics and its environment, the network has to estimate the unknown information during its operation. If this estimated information gradually approaches the true information as time proceeds, then the network approaches that of an optimal controller. Such a controller can be viewed as an adaptive controller, due to the gradual improvement of the estimated information.

![Typical Neural Learning Scheme](image)

**Figure 2.12 Typical Neural Learning Scheme**
The controller learns the unknown information during operation, and this information, in turn, is used as an experience for future decision and controls. A control system is called a **learning control system**, if the information pertaining to the unknown features of the plant or its environment is acquired during operation, and the obtained information is used for future estimation, recognition, classification and control or decision so that the overall system performance is improved. Once learning is complete, the control system can compensate for large number of changes in the plant and its environmental conditions. The difference between adaptive and learning system lies in the fact that the former treats every distinct operating situation as novel, whereas the latter correlates the past experience with the present situations and accordingly adapts its behavior.

### 2.8.2 ANN for system identification

An obvious approach for system modeling is to choose the input-output structure of the neural network to be the same as that of the system. If the output of the network is defined as $y$ then the system relationship can be expressed as $y(k)=f(y^p(k-1),...,y^p(k-n); u(k),...,u(k-m))$. It is a well established fact that a feed forward network of the multilayer perceptron type can approximate arbitrarily well a continuous function (Cybenko 1988, Funahashi 1989). The different types of neural network architectures used in system identification are (i) forward modeling (ii) inverse modeling (iii) model reference control (iv) internal model control (v) optimal decision control and (vi) predictive control.
2.8.2.1 Forward modeling

The neural network is trained to represent the forward dynamics of a system. The ANN model is placed in parallel with the system (Jordan et al. 1991) and the error, (the prediction error) between the system and network outputs is used as the network training signal as shown in Figure 2.13. To train the network, target values have to be provided directly in the output coordinate system of the learner.

![Figure 2.13 Forward modeling neural network](image)

2.8.2.2 Inverse modeling

The direct inverse model structure is shown in Figure 2.14 (Psaltis et al. 1988). In this a synthetic training signal is introduced into the system. The system output is then used as input to the network. The network output is
compared with the training signal and this error is used to train the network. This method has, however, the drawback of not being "goal directed" (Jordan and Rumelhart 1991). The inverse model is cascaded with the controlled system resulting in an identity mapping between desired response and the controlled system output. This model has the drawback that if the nonlinear system mapping is not one-one then an incorrect inverse may be obtained.

Figure 2.14 Inverse model neural network (Direct Method).

2.8.2.3 Specialised inverse learning

In this approach (Psaltis et al.1988) the network inverse model precedes the system and receives as input a training signal which spans the desired operational output space of the controlled system. The learning structure also contains a trained forward system model placed in parallel with the plant. This is shown in Figure 2.15.
Figure 2.15 Specialized Inverse learning neural network

2.8.2.4 Model reference control

In this, the desired closed-loop system performance is specified through a stable reference model $M$, which is defined by its input-output pair \{$r(t), y'(t)$\}. The objective of the control system is to make the plant output $y^p(t)$ match the reference model output asymptotically, i.e. $y'(t)$ satisfy the condition $\lim_{t\to\infty} ||y'(t)-y^m(t)|| < \varepsilon$ for some specified constant $\varepsilon \geq 0$. This control structure is shown in Figure 2.16 (Narendra and Parthasarathy 1990).
2.8.2.5 Internal model control

In this, a system forward and inverse model is used directly as elements within the feedback loop (Morari et al 1989). This structure extends readily to non-linear control. A system model is placed in parallel with the real system. The difference between the system and the model is used for feedback purposes as shown in Figure 2.17.
2.8.2.6 Predictive Control

This is shown in Figure 2.18 and provides for a prediction of the future plant response over the specified horizon (Keerthi et al 1990). The predictions made by the network can be used for optimizing or minimizing a specified performance criterion in the calculation of a suitable control signal. The advantage is that, the optimization routine is no longer needed when training is complete.

![Predictive Control Structure](image)

**Figure 2.18 Predictive Control Structure**

2.8.2.7 Optimal Decision Control

In this, the state space is partitioned into regions (feature space) corresponding to various control situations (pattern classes). The realization of the control surface is accomplished through a training procedure. Since the
time-optimal surface is nonlinear, it is necessary to use an architecture capable of approximating such a surface. The optimal control structure using neural network is shown in Figure 2.19. In this structure, the state space is first quantized into elementary hypercubes, in which the control action is assumed constant. This is followed by another network acting as a classifier which can be implemented with a standard back-propagation architecture.

![Diagram](image)

**Figure 2.19 Structure for optimal control**

### 2.8.3 Two Layered Feed Forward Neural Network

A layered feedforward neural network consists of Adalines connected together as shown in Figure 2.20. A layer of Adalines is created by connecting a number of Adalines to the same input vector. Many such layers can be cascaded, with output of one layer connected to the inputs of the next layer, to form a network. It has been proven that a network consisting of only two layers of Adaline can implement any nonlinear function \( X, d(X) \) given enough
Adalines in the first layer (Miyake et al. 1988). The idea is that each Adaline in the first layer can take a small piece of the function relating $X$ to $d(X)$ and make a linear approximation to that piece. The second layer then adds the pieces together to form a complete approximation to the desired function.

2.8.4 Independent Component Analysis

Independent Component Analysis (ICA) is an unsupervised learning technique that in many cases characterizes the data in a natural way. The main application area of ICA is the blind signal separation (BSS). In signal separation, multiple streams of information are extracted from linear mixtures of these signal streams. This process is blind, if examples of the source signals,
along with their corresponding mixtures, are not available for training. In BSS, signals are estimated from their unknown linear mixtures with the assumption that the sources are independent.

### 2.8.4.1 ICA Data Model

The ICA operates on \( M \) zero mean source signals \( s_k(1), \ldots, s_k(m) \), \( k=1,2,\ldots \), that are scalar-valued and mutually statistically independent for each sample value \( k \). The original sources are unobservable and the input to the ICA are different linear mixtures \( x_k(1), \ldots, x_k(L) \) of the sources. The ICA model in vector form is given as

\[
M \\
\mathbf{x}_k = \mathbf{A}^* \mathbf{s}_k = \sum_{i=1}^{M} s_k(i) \mathbf{a}(i)
\]  

(2.1)

where \( \mathbf{s}_k = [s_k(1), \ldots, s_k(M)]^T \) represents the source vector of the \( M \) source signals \( s_k(i) \) (\( i=1,\ldots,M \)) at the index value \( k \) and \( \mathbf{A} = [\mathbf{a}(1), \ldots, \mathbf{a}(M)] \) is a constant \( L \times M \) mixing matrix whose elements are the unknown coefficients of the mixtures. The columns \( \mathbf{a}(i) \) are the basis vectors of ICA. Usually, the number of source \( M \) is assumed to be equal to the number of available different mixtures \( L \), to simplify the derivation of BSS algorithms. However, in practice, it is not necessary for \( M \) to be equal to \( L \).

### 2.8.4.2 Blind Source Separation

The structure of BSS for instantaneously mixed sources (Yang 1999) is shown in Figure 2.21. In BSS, the objective is to separate mutually statistically independent but otherwise known source signals from
their linear mixtures without knowing the mixing coefficients. The task is to find individual source signals \( \{s_k(i)\} \), with only the data vectors \( x_k \) and the number of sources \( M \) known.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{ICA_network.png}
\caption{ICA network}
\end{figure}

### 2.8.4.3 BSS Algorithm

There are two types of BSS problems: those that involve instantaneous mixtures and those involving convolutive mixtures. In the present work, the BSS separation algorithm is considered for problems involving instantaneous mixtures. The ICA architecture used to perform the separation of source signals and to estimate basis vectors consists of (i) whitening (ii) separation and (iii) estimation layers. The whitening process is applied to the input vectors so that (i) the data vectors have a zero mean (ii) the variances of the observed signals are normalized to unity and (iii) the separation algorithms have better stability properties and converge faster. The components
of the whitened vectors are mutually uncorrelated since it is a necessary prerequisite for the stronger independence condition.

The separation process can be modelled as a single layer neural network with equal number of input and output nodes, as shown in Figure 2.21 where the coefficients $w_{ij}$ of the separation matrix $W$ are simply the weights from input to output nodes. The activation function at the output nodes is used for the training mode only. The threshold activation function plays a central role in blind signal separation and is discussed in detail in a later chapter.

### 2.8.5 Back propagation Algorithm

The governing equations for a back propagation net, (Rumelhart, et al 1986), such as in Figure 2.22 are briefly reviewed here. The neurons in the input layer simply store the input values. The hidden layer and the output layer neurons each perform two calculations. First, they multiply all inputs with weights and add the bias to form the sum $S_j$

$$S_j = \sum_{i=1}^{N} w_{ij} x_i + b_j$$  \hspace{1cm} (2.2)
Second, the output of the neuron, \( O_j \), is calculated using the sigma function of \( S_j \) as

\[
O_j = \sigma (S_j) \quad (2.3)
\]

where \( \sigma (z) = [1 + \exp (-z)]^{-1} \quad (2.4) \)

It is not necessary for all of the nets to use the sigmoidal function given in equation (2.4), but they are the ones commonly used. A backpropagation net learns by making changes in its weights in a direction as to minimize the sum of squared errors between its predictions and a training data set. The minimization is done using algorithms such as the steepest descent or the conjugate gradient or Newton’s method. If there are \( R \) input-output pairs,
\( x^{(r)} \), \( y^{(r)} \) available for training the net, then after presentation of a pair \( r \), the weights are changed as follows:

\[
\mathbf{w}_{iv}^{(r)} = \mathbf{w}_{iv}^{(r-1)} + \Delta \mathbf{w}_{iv}^{(r)} \tag{2.5}
\]

with \( \Delta \mathbf{w}_{iv}^{(r)} \) given by

Hidden layer to output weights

\[
\Delta \mathbf{w}_{ik}^{(r)} = \sigma_{ik} (S_k) \left[ y^{(r)} - O_{ik}^{(r)} \right] O_j \tag{2.6}
\]

Input to hidden layer weights

\[
\Delta \mathbf{w}_{ij}^{(r)} = \sigma_j (S_j) \left( \sum_k \left[ \sigma^* (S_k) \right] \left[ y_k^{(r)} - O_k^{(r)} \right] w^{(r-1)} \right) x^{(r)} \tag{2.7}
\]

and

\[
\sigma^* (S_k) = \sigma (S_k) \left[ 1 - \sigma (S_k) \right] \tag{2.8}
\]

The weights are changed with presentation of each pair.

One might assume that setting all the weights to zero may be an acceptable starting point. If all the weights start with equal values and the solution requires unequal weights to be developed, then the system can never learn. The reason is that, the error is propagated back through the neurons in proportion to the value of the weights as shown by equation 2.7. All the error signals to the hidden nodes remain identical, and the system starts at a local minima and remains there. Starting the system with a set of randomized weights distributed uniformly between -0.5 and +0.5 counteracts this problem. The
chosen activation function has a special feature. The function cannot reach its final values 0 and 1 without infinitely large inputs. The useful region of the activation is approximately between 0.1 and 0.9 and the variable’s input to the net is scaled within this range.

2.9 SUMMARY OF COMMONLY EMPLOYED NEURO MODELS

<table>
<thead>
<tr>
<th>Name of network</th>
<th>Primary applications</th>
<th>Limitations</th>
<th>Comments</th>
</tr>
</thead>
</table>
| Adaptive Resonance Theory        | Pattern recognition, especially when pattern is complicated.              | (i) Sensitive to translation  
(ii) distorition, changes in scale | Very sophisticated, not yet applied to many problems.                      |
| Avalanche                        | Continuous-speech recognition, teaching motor commands to robotic arms    | (i) Literal playback of motor sequences  
(ii) no simple way to alter speed/interpolate movements | Class of networks. No single network can do all these tasks.              |
| Back propagation                 | Speech synthesis from text, adaptive control of robotic arms.             | (i) Supervised training only  
(ii) abundant input-output patterns.                                        | The most popular network today-works well, simple to learn                |
| Bi-directional associative memory| Content-addressable associative memory                                   | (i) Low storage density  
(ii) data must be properly coded                                             | Associates fragmented pairs of objects with complete pairs                |
| Boltzmann and Cauchy machines    | Pattern recognition for images, sonar, radar                              | (i) Boltzmann machine: long training time  
(ii) Cauchy machine: generating noise in proper statistical distribution | Simple networks in which noise function is used to find a global minimum  |
| Brain state in a box             | Extraction of knowledge from data bases                                   | (i) One-shot decision making  
(ii) no iterative reasoning                                                      | Similar to bi-directional associative memory in completing fragmented inputs |
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cerebellatron</td>
<td>Controlling motor action of robotic arms</td>
<td>Requires complicated control output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Similar to avalanche network; can blend several command sequences with different weights to interpolate late motions smoothly.</td>
</tr>
<tr>
<td>Counter-Propagation</td>
<td>Image compression; statistical analysis.</td>
<td>Large number of processing elements and connections are required.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Functions as a self-programming look-up table similar to back propagation.</td>
</tr>
<tr>
<td>Hopfield</td>
<td>Retrieval of complete data or images from fragments.</td>
<td>Does not learn-weights. They must be set in advance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Can be implemented on a large scale.</td>
</tr>
<tr>
<td>Madaline</td>
<td>Adaptive nulling of radar jammers, adaptive modems, adaptive equalizers in telephone lines</td>
<td>Assumes a linear relationship between input and output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Acronym stands for multiple adaptive linear elements, powerful learning law, in commercial use for more than 20 years</td>
</tr>
<tr>
<td>Necognitron</td>
<td>Handprinted-character recognition</td>
<td>Requires unusually large number of processing elements and connections</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Most complicated network ever developed, insensitive to differences in scale, translation, rotation; able to identify complex characters (such as Chinese)</td>
</tr>
<tr>
<td>Perceptron</td>
<td>Typed-character recognition</td>
<td>(i) Cannot recognize complex characters (such as Chinese)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ii) Sensitive to difference in scale, translation, distortion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The oldest neural network known; was built in hardware; rarely used today</td>
</tr>
<tr>
<td>Self-organizing map</td>
<td>Maps one geometrical recognition (such as a rectangular grid) onto another (such as an aircraft)</td>
<td>Requires extensive training</td>
</tr>
<tr>
<td></td>
<td></td>
<td>More effective than many algorithmic techniques for numerical aerodynamic flow calculations</td>
</tr>
</tbody>
</table>