vehicles. The equations of motion for the AUV used in the studies are summarized in Appendix C. In all the chapters, figures and tables are shown along with the appropriate sections in the text.

mance compared with the other schemes[53][54][65][66]. It is also shown that adding a velocity reference signal improves the dynamic response of the neural network controllers[64][67].

3. A number of new applications are introduced for the neural networks. These include on–line learning control of autonomous underwater vehicles(using feedforward as well as recurrent network architectures)[53][54]56][62][65][66], continuous recognition of sonar targets[59][60] and 2D object recognition using Fourier Descriptors[57][63].

1.4. ORGANIZATION OF THE DISSERTATION

The dissertation is organized into eight chapters. Following the introduction chapter (Chapter 1), the second chapter gives an overview of the connectionist learning procedures and different architectures used in the applications involving dynamical systems. The Alopex algorithm and the results of experimental studies on a number of benchmark problems are given in Chapter 3. The different direct control schemes are reviewed and analyzed in Chapter 4. The proposed gain layer scheme and the results of comparison with the existing schemes are also furnished in the same chapter. The results of studies on gain layer scheme, using linear as well as nonlinear dynamical models of the autonomous underwater vehicle are continued in Chapter 5. Chapter 6 provides a comparative overview of important recurrent neural network architectures. Chapter 7 presents the results of studies on Alopex algorithm as a learning algorithm for recurrent networks. A summary of the research and concluding remarks are provided in Chapter 8. Appendix A gives relevant details on the Fourier Descriptors, which are used in the character recognition studies described in Chapter 3. Appendix B describes some of the control approaches currently used in autonomous underwater

formance of the controller; and, (iii) The ability of the network controller to adapt to sudden as well as slow parameter variations in the dynamics.

In the final set of experiments, the effectiveness of the Alopex on recurrent networks with asymmetric feedback connections is investigated. The existing recurrent network architectures and algorithms for applications involving dynamical systems are reviewed. The Alopex algorithm is then presented as an appropriate learning algorithm for some of these architectures. Using the AUV control problem, it is shown that recurrent neural networks trained using the Alopex algorithm provides good tracking performance and stability. It is further shown that specific improvements in the network complexity could be achieved using the present scheme. It is also indicated that a trained Alopex neural network has better measurement noise rejection capabilities than the backpropagation algorithm over a large range of signal–to–noise ratio(SNR).

1.3. CONTRIBUTIONS

Following are the specific contributions resulting from this research:

1. The Alopex algorithm is introduced as a universal learning algorithm for the neural networks. It is shown that the algorithm could be effectively used as a learning algo–rithm on a wide variety of network architectures and on a number of diverse applica–tions in static classification, temporal pattern recognition and learning control. It is also shown that the algorithm has specific advantages over backpropagation in a number of problems and network architectures[50][55–62].

2. An improved direct control scheme is developed for neural network applications. It is shown that the present scheme gives better dynamic response and tracking perfor–

ctral temporal dependencies of the consecutive input patterns. A time–delay network architecture[47][48] is employed for the recognition. These ideas are further extended in the second experiment for recognizing the temporal XOR signals[49][50]. A recurrent network architecture employed in this study shows the effectiveness of Alopex algorithm on networks with feedback connections. It may be noted that many of the popular learning algorithms cannot be directly used on recurrent networks or are computationally too intensive for such tasks.

The static recognition tasks can provide some useful insight into the performance of different learning algorithms in a dynamical environment. In order to deal with dynamical classification problems elegantly, the network has to learn temporal dependencies of consecutive input patterns. In temporal recognition tasks, the network has to keep track of the information about previous states of the neurons as well as the inputs. The sonar target recognition problem discussed earlier is an example for such a situation.

In the next set of experiments, the temporal pattern classification studies are extended to learning control applications. The essential similarities and differences between the pattern classification and learning control processes are reviewed and the application of the Alopex algorithm on on–line learning control tasks is investigated. The direct control architecture is used in the relevant studies. Note that the backpropagation algorithm cannot be used as such on direct control architectures[51][52]. Two on–line direct control schemes viz, Gain-layer schemes[53][54] which perform better than the other existing architectures, are proposed. Detailed studies are conducted on the gain layer schemes using linearized as well as nonlinear dynamical models of an Autonomous Underwater Vehicle(AUV). Three specific issues addressed in the studies are: (i) The complexity of the network, in terms of the number of interconnection strengths, with linear and nonlinear dynamics; (ii) The effect of network parameters such as the learning rate and the initial weight distribution, on the tracking per-

on a few popular learning algorithms, such as backpropagation. Hence, warranted is a new recourse in investigating alternate learning algorithms. As such, the objective of the current research is to investigate the Alopex algorithm as a universal learning algorithm for artificial neural networks, and is studied on a number of architectures including recurrent networks as well as on a wide variety of benchmark problems and in complex practical applications. Emphasis is placed on using the algorithm on time–varying signals and systems; for example prediction of continuous time signals and modeling and control of complex, nonlinear dynamical systems.

1.2. THE RESEARCH APPROACH

A schematic of the approach advocated in the present studies is shown in figure 1.1. The Alopex algorithm is studied on a variety of problems with necessary modifications. Its effectiveness as a learning algorithm is first investigated on two different classes of problems through simulation studies, namely (a) learning and classification of static input patterns; and, (b) learning and classification of dynamic input patterns. The algorithm is envisaged on two practical problems, each corresponding to the two classes of aforesaid problems.

In the static supervised pattern classification efforts, the algorithm is studied on a number of standard benchmark problems such as XOR, Parity, Symmetry and MONKS problems. The Encoder problems of different dimensions are studied in order to investigate the scaling properties of the Alopex algorithm. In all the above studies, the algorithm is compared with the backpropagation algorithm in terms of the required number of iterations. The performance of the Alopex algorithm in practical classification tasks is demonstrated using a handwritten digit recognition problem.

The second set of studies consists of two simulation experiments. In the first experiment for recognizing undersea targets from side–scan sonar returns, the network has to learn the spe–

In order to find the true global minimum, the network has to adjust its parameters so as to move in a direction opposite to the error gradient from time to time. This is generally called the *Monte Carlo* algorithm and was introduced by Metropolis *et al.* in 1953[36]. This idea was later extended as an efficient optimization process (simulated annealing) by Kirkpatrick *et al.*[37] and Geman *et al.*[147]. It was shown by Hinton *et al.*[14][31] and Ackley *et al.*[33] that networks which converge to the global minimum could be formulated by combining the Hopfield network and the simulated annealing algorithm. Their network model (called the Boltzmann machine), was found to solve a number of constraint optimization problems[34]. The major drawback of the Boltzmann machine learning scheme is that it is restricted to the Hopfield type networks with neuron states operating in the binary mode, in confirmation with McCulloch–Pitts *'all–or–none'* response of neurons.

Harth *et al.*[38][39] and Pandya[40] showed that an efficient optimization algorithm could be formulated by adding stochasticity into the Alopex* algorithm[41][42][43]. This formulation was able to solve a number of optimization problems such as Travelling Salesman Problem(TSP), Diet Problem and Crystal Growth[40]. The Alopex algorithm had also been used earlier on a number of studies on visual receptive fields[41][43] and role of feedback in visual perception[42][44]. Recent studies have also reaffirmed that the Alopex process is plausible biologically[45]. The objective of this dissertation is to investigate rigorously the Alopex algorithm as a learning algorithm for neural networks.

As the problems in which neural networks could be used effectively, are expanding and hardware implementations of the neural networks are realized in VLSI forms, a relevant area of research will be to develop a learning algorithm independent of the architecture and is powerful in other aspects such as computation, memory, and simplicity. As Lipman[46] points out, a major shortcoming of much of the current neural network research is the overemphasis

* Alopex is the acronym for Algorithm for Pattern Extraction

Hence, choosing an architecture is very much problem dependent. Since most of the present learning algorithms are designed for a specific architecture, choosing an architecture means choosing the learning algorithm concurrently. A multilayer perceptron learning algorithm like backpropagation cannot be used directly as such on a recurrent network architecture. The algorithms such as Boltzmann learning and A_{R-P} could be used only for specific architectures. The basic reason behind this handicap is that the learning algorithm is not a separate entity but, it is an essential part of the architecture itself. The flexibility of the learning algorithm is also affected, by the specific class of error functions and the neuronal transformation. In this dissertation an approach is presented which alleviate these difficulties.

1.1. RATIONALE OF THE PRESENT RESEARCH

Over the past few years backpropagation has become the most popular learning algorithm for the neural networks. As Hinton[31] points out, in spite of its impressive performance on relatively small problems and its promise as a widely applicable learning algorithm for extracting underlying structures of a domain, backpropagation is rather inadequate for larger learning tasks. The learning time for backpropagation scales poorly as the learning task becomes more complex. Also, it is interesting to note that this most popular learning algorithm for artificial neural networks (backpropagation) is not plausible biologically. There is no evidence that biological synapses can be used in the reverse direction to propagate the error or or that neurons can propagate error derivatives[31][34].

The general drawbacks of the gradient decent approaches apply to the backpropagation algorithm also. The algorithm is sensitive to initial weight distribution[35] and may converge to a local minimum rather than to a global minimum[14][31][34]. Further, backpropagation requires that[14] the nonlinear neuronal transformation to be a monotonically increasing and differentiable function. Also, existence of the derivative of the error function is a necessary requirement.

perceptrons. The work by Rumelhart *et al.* helped in reviving the field of artificial neural networks. A number of immediate applications were found for such networks in engineering, computer science, economics and physical sciences.

Some of the other implications of the research by Rumelhart *et al.* and the PDP Research Group[14] were that a number of other learning algorithms and architectures, some of them developed much earlier than the publication of their results, came to be noticed. These include the Adaptive Resonance Theory (ART) by Grossberg[15][16], the Reinforcement learning and Associative Reward and Punishment (A_{R-P}) by Barto *et al.*[17–22], the Cerebellar Model Articulate Controller(CMAC) by Albus[23–26], Self–organizing networks by Kohonen[27][28] and Neocognitron by Fukushima[29][30], and the Boltzmann machine[31–33].

A number of interesting applications were found for the artificial neural networks in the subsequent years. They include pattern classification tasks[13][18–20][46–77], learning control of dynamical systems[24–26][148–161][163–167], pattern retrieval systems[79–82], financial market prediction and defense applications[148]. It is not an exaggeration to say that, every field in engineering has sought virtually seeked the neural network approach to solve a number of complex problems. In most of these applications, the neural networks are used as the pattern classifiers where the input patterns are static in nature. There are specific learning and classification stages in these applications. Studies on more complex tasks such as sequence prediction, speech recognition and learning control, show that neural networks offer promising solutions to a variety of problems in these fields also.

The architecture of the networks play an important role in any such application. For example a Hopfield type network is more suitable as a content addressable memory rather than as a signal prediction network. A recurrent network with asymmetric connections may be more appropriate than a feedforward network, in system identification and control applications.

McCulloch and Pitts. The synaptic strengths in the network were adapted according to a learning rule, to minimize the error between the desired and actual responses. Shortly after Rosenblatt's publication about the 'perceptrons', Widrow and Hoff[5] proposed a single layer network called Adaline, which differed from the perceptrons only in its learning procedure. The above developments are generally referred to connectionist models, meaning that the information storage in the brain is in the form of synaptic weights which are adapted during the learning phase.

The initial excitement due to the above discoveries in modeling the biological neural systems however soon died out as it was shown by Minski and Papert[6] that perceptrons or Adalines are unable to perform even elementary logic operations such as XOR. Even though a small number of researchers such as Amari, Fukushima, Cooper and Grossberg still continued research on modeling the behavior of biological neurons and developing mathematics and architectures for extracting features and patterns, the modern revival of the connectionist research started with the appearance of two seminal publications by Hopfield[7][8] on neuron–like elements with symmetric connections between them. Hopfield on the basis of the analogy between the spin–glass models[9] and neural networks advocated by Cragg & Temperly[10], developed his networks, which are generally known as Hopfield networks. He showed how such networks can be effectively used as Content Addressable Memories (CAM). Even though the Hopfield networks are not established to be biologically plausible, they have been useful in solving some interesting problems such as Traveling Salesman Problem (TSP)[11], task assignment problems[12] and image labelling[13].

In 1986, Rumelhart *et al.*[14] introduced a learning algorithm called the error backpropagation which could be used for training perceptrons of more than one layer of weights. Multilayer architectures were able to form complex decision regions, there by enabling the network to solve problems, such as XOR, which were once considered as unsolvable using

CHAPTER 1.

INTRODUCTION

Biological information processing systems are some of the most highly organized structures. They are compact, energy efficient and self adaptive systems, which can monitor and control all the body–functions in an optimal way. They outperform the conventional digital computers in a number of functionalities such as perception, optimization, learning and control. A wide range of theoretical as well as experimental studies have been conducted to model the brain. Even though Cajal, in 1908, first identified that[1] neurons are the basic cellular constituents in the brain, the first major contribution in terms of mathematical modeling the network of neurons was presented only in 1943 by McCulloch and Pitts[2]. The concept of learning in brain was proposed by Hebb in 1949, suggesting that the connectivity in the brain is changing continually as the organism learns different functional tasks[3].

In 1958, Rosenblatt[4] derived a neural network model called 'Perceptrons', which could learn to classify certain sets of patterns, as similar or distinct. The network consisted of a single layer of neurons, with each neuron possessing the characteristics proposed earlier by

sented by equation (2.9). The generation of the control signal *u(k)* in such a case using indirect control scheme is difficult as one has to find the inverse of the function *f(.)* and has to worry about its existence. Also, since the actual control signal is estimated algebraically rather than via direct adapting rule, it could be argued that the indirect control scheme shown in figure 2.8b does not fully represent a neural network approach.

In the direct control scheme, as such there is no identification process and hence, the algebraic estimation of the control signal is not necessary. The neural network is adapted at each instant to minimize the system error *e*. Since an exact inverse of the plant is not identified, the approach may work even for the plants whose inverse is not well defined[26]. The major difficulty in the direct control approach is the estimation of the error at the output of the network, which is required for estimating the error gradients to update the weights. Since the plant lies between the controller and the output error, the error *e* does not represent the error at the output of the network. Existing approaches to overcome these problems and some of the proposed modifications are discussed in detail in Chapter 4. Note that in the case of the Alopex algorithm, since an exact gradient of the error is not calculated, the controller could be trained using the total error *e* itself.

1.4. SUMMARY

In this chapter the neural network approaches to pattern classification and control are reviewed. The different learning algorithms and their performance characteristics as pattern classifier algorithms are explained. The direct and indirect schemes for the neural network based adaptive control of dynamical systems are presented with some examples. A more detailed review and analysis of some of the relevant approaches related to the current study are given in the subsequent chapters at appropriate sections.

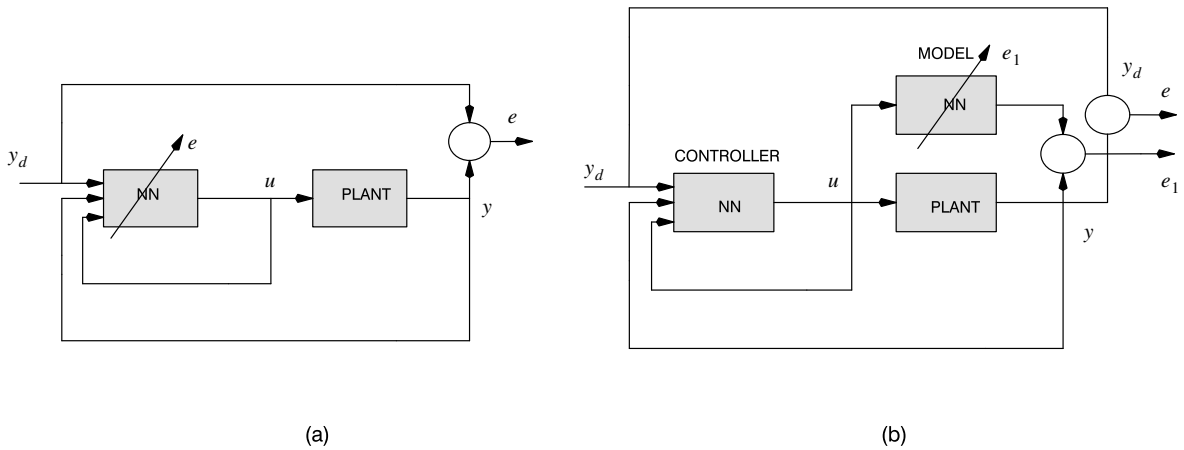


Figure 2.8 Adaptive control techniques for dynamical systems
(a) Direct Control
(b) Indirect Control

As seen from the figure, the direct control scheme is simpler than the indirect scheme. The indirect control scheme requires algebraic determination of the control signal depending upon the type of the plant. This could be explained more explicitly by considering an example. Consider a dynamical system described by[51].

$$y(k+1) = \frac{5y(k)y(k-1)}{1 + y^2(k) + y^2(k-1) + y^2(k-2)} + u(k) + 0.8u(k-1) \qquad (2.12)$$

Since the desired output of the overall system y_d is known, the control signal *u(k)* at any instant *k* could be estimated as follows:

$$u(k) = -\hat{f}[y(k), y(k-1), y(k-2)] - 0.8u(k-1) + y_d(k) \qquad (2.13)$$

The nonlinearity part of the equation (2.12) is estimated using a neural network as shown in figure 2.8b. Such an approach is easily practicable as long as the input signal *u(k)* is easily separable from the equations representing the dynamics. Consider a dynamical system repre-

1.3.2. Control

According to conventional adaptive control theory, the controller design process can be split in to two parts[51][85]. In the first part, referred to as the algebraic part, it is checked whether the controller has necessary degrees of freedom to produce the desired control signal. In other words, it is verified that the controller parameter vector exists for every value of the plant parameter vector. In the second part, called the analytical part, stable adaptive laws are determined for adjusting the initial controller parameters $\theta(k)$ so that $\lim_{k\to\infty} \theta(k) = \theta^*$ and the error *e(k)* at the output tends to zero. The approach remains the same for a neural network based control scheme. Here the objective is to adjust the network weights so that the system error *e(k)* tends to zero asymptotically.

The control action can be either off–line or on–line. In the off–line scheme, a trained neural network is used as the controller and there is no learning involved during the control. For such a scheme, input/output data of the system is needed *a priori*. In the on–line control scheme there is no *a priori* training involved. The network is started from random initial conditions and the learning continues as long as the control signal is needed. The studies described in this dissertation concentrate only on the on–line learning control scheme.

There are two distinct approaches to control a system adaptively namely, (a) direct control and (b) indirect control[85]. Figure 2.8 illustrates the approaches. In the indirect control, the parameters of the plant are estimated as the elements of a vector $\hat{p}(k)$ at each instant k and the controller parameters $\theta(k)$ are estimated assuming that $\hat{p}(k)$ represents the true parameters $p(k)$ of the plant. There is an explicit identification process in this approach. In the direct control approach, the parameters of the controller are directly adjusted to reduce the output error. It is to be noted that even when the plant is assumed to be linear and time–invariant, both the approaches result in overall nonlinear systems.

dom input signal uniformly distributed between –1.0 and +1.0 and the weights of the networks are adapted for 50, 000 iterations. The figure compares the output of the nonlinear function and the network for an input signal $u(k) = \sin(2\pi k/250)$, after the identification. The results confirm that the network is able to model the system well. A similar result for a dynamical nonlinear system of the form,

$$f[y(k), y(k-1)] = \frac{y(k)y(k-1)[y(k)+2.5]}{1+y^2(k)+y^2(k-1)} \tag{2.11}$$

is shown in figure 2.7b. In this case, the number of learning cycles needed is 125, 000 which is much higher than that required for identifying the static nonlinearity.

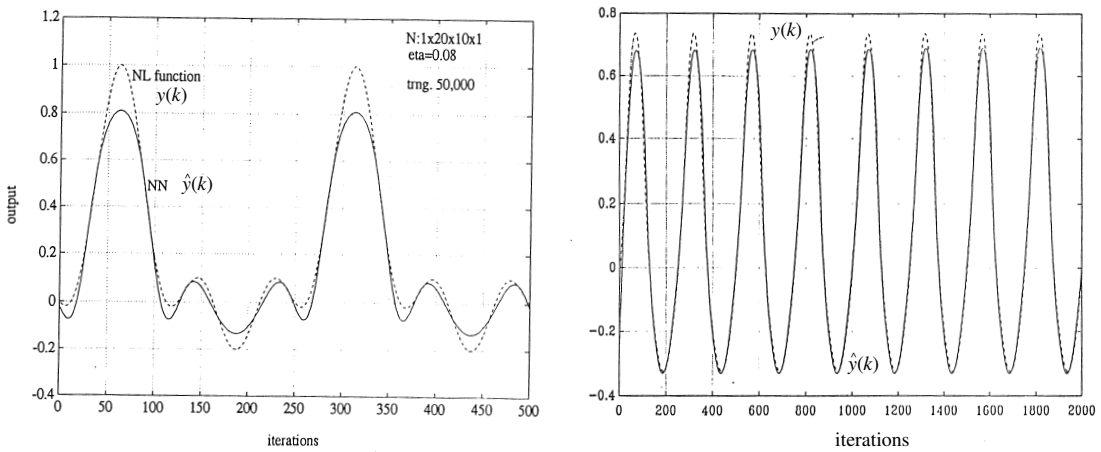


Figure 2.7 Identification of dynamical systems using neural networks

(a) Identification of a static nonlinear system using parallel scheme

(b) Identification of a nonlinear dynamic system using series parallel scheme

of the plant, rather than the output of the model itself, is fedback as the input to the neural network.

Consider a nonlinear dynamical system represented by the following equation:

$$y(k+1) = f[y(k), y(k-1), ..., y(k-n+1); u(k), u(k-1), ..., u(k-m+1)] \qquad (2.9)$$

The output of the dynamical system depends on n previous values of the output and m previous values of the input. If we model the system using a feedforward network, using the series–parallel model, the network has to have the signals $y(k), y(k-1), ..., y(k-n+1)$ and $u(k), u(k-1), ..., u(k-m+1)$ as the input. In a recurrent network based identification scheme, only *y(k)* and *u(k)* may be needed as the inputs. For effective identification of the plant over a wide range of input signals a uniformly distributed random signal is chosen as the input signal *u(k)*.

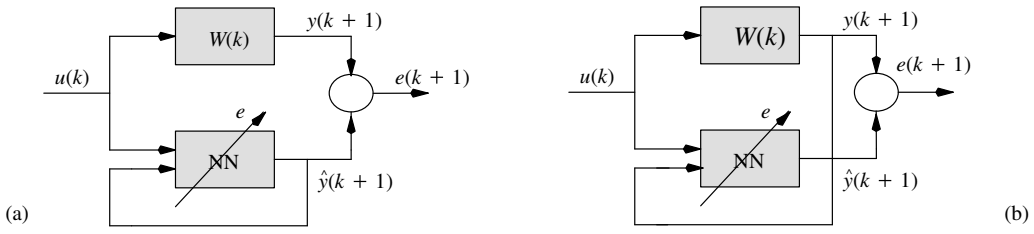


Fig. 2.6 Identification of dynamical systems using neural networks

(a) Parallel Identification
(b) Series–Parallel Identification

Figure 2.7a. shows the results of identification of a static nonlinear function,

$$f(u) = 0.8u^3 + 0.4u^2 - 0.2u \qquad (2.10),$$

using a feedforward backpropagation neural network. The identification is done using a ran-

tion. The examples are Learning Vector Quantization (LVQ) networks[27][28] and Adaptive Resonance Theory(ART) networks[15].

A comparison of different neural network approaches to classification in terms of memory requirements and training time is given in figure 2.5.

1.3. NEURAL NETWORKS FOR IDENTIFICATION AND CONTROL

Two of the most important properties of the neural networks in relevance to their applications in control systems are the approximation capability and learning ability. The development of appropriate supervised training algorithms and architectures to efficiently solve function approximation problems, has received considerable attention in recent years[81][82]. Multilayer networks with at least one hidden layer of weights and sigmoidal transformation functions, can approximate large classes of nonlinearities[83][84]. From a control system point of view feedforward neural networks represent static nonlinear maps and recurrent networks represent dynamic feedback systems[51]. Like any traditional parameter estimation process, the network parameters could be learned to model a function or a system.

1.3.1. System Identification

System identification consists of setting up a suitable parameterized model and adjusting the parameters to optimize a performance function based on the error between outputs of the plant and the model. Figure 2.6 shows two traditional identification schemes. In the parallel identification scheme, the network has no direct information about the plant outputs, other than its own outputs and inputs. The stability of the plant does not guarantee the stability of the identification model in such an approach[51]. Such a scheme is used normally only for identification of static nonlinearities. In the series–parallel identification scheme, the output

the width of the kernel function. The function provides the strongest output when the input is near the node centroid. Receptive field classifiers train relatively fast and have intermediate memory and computation requirements. Examples are the Cerebellar Model Articulate Controller (CMAC) network classifiers[23] and radial basis function classifiers[73]. It is worth noting here that, in recent years, the radial basis function network as well as CMAC networks have shown good potential in control applications[24–26][80].

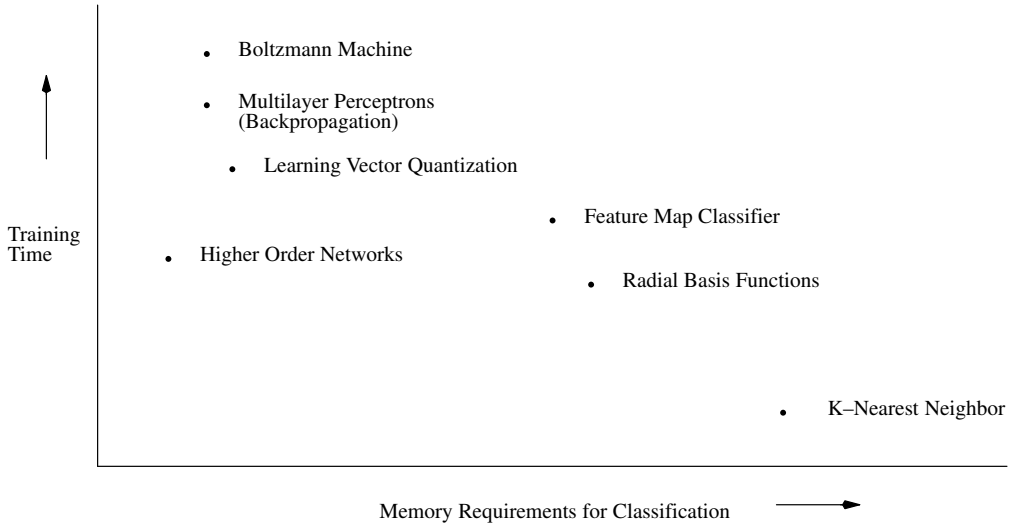


Figure 2.5 Relative difference between classifier training time and memory requirements(adopted from [46])

Exemplar classifiers perform classification based on the identity of the training examples that are nearest to the input, or nearest neighbors, which are determined using exemplar nodes that are similar to the kernel function nodes in equation (2.8). Exemplar nodes compute the weighted Euclidean distance between the inputs and the node centroids. The node centroids are determined during the learning phase from the training samples. Such classifiers train rapidly but require large amounts of memory and computation time for classifica-

Based on the way decision regions are created, the family of pattern classifiers could be divided into four broad groups as shown in table 1.1[46]. The probabilistic classifiers assume *a priori* probability distribution such as Gaussian for input features. Parameters of the distributions are estimated typically using a supervised training scheme. These classifiers provide optimal performance when sufficient training data is available to learn the parameters accurately so that the distributions are true models of the test data. The hyperplane classifiers use the sigmoidal nonlinearity as the computing element. These classifiers have low memory and computation needs, but may require long training time[46]. The examples of such classifiers are, multilayer perceptrons(MLP) trained using backpropagation algorithm, Boltzmann machine classifiers, binary tree classifiers[79] and higher order networks[79].

Table 1.1 The Four Different Groups of Classifiers

GROUP	COMPUTING ELEMENT	CLASSIFIER
Probabilistic	Distribution dependent	Gaussian or Mixture
Hyperplane	Sigmoid	MLP, Boltzmann Machine
Receptive field, Radial Basis	Kernel or radial Basis	CMAC, Radial Basis Networks
Exemplar	Euclidian Norm	K–Nearest Neighbor Linear Vector Quantization(LVQ)

Receptive field classifiers create complex decision regions that form overlapping receptive fields. In this case, the neuronal transformation function is of the form:

$$o_j = f_k(\| x - m_j \| / h) \tag{2.8}$$

where f_k is the kernel function. The parameter m_j is the centroid of node j and h determines

The probability of finding such a system in any global equilibrium, obeys the Boltzmann distribution.

At high temperatures, the network approaches the equilibrium rapidly but low energy states are less probable than high energy states. At low temperatures the network reaches equilibrium slowly and low energy states are more probable than high energy ones. The fastest way to approach the low temperature equilibrium is to start generally from a higher temperature and reduce the temperature gradually. This procedure is generally termed as the simulated annealing. A similar approach for the Alopex algorithm is discussed in Chapter 3.

The training of the network is done[33] by clamping the input and output vectors to the training input vector and desired output vector respectively. The network is then allowed to anneal until it approaches a thermal equilibrium at a temperature of '*1*'. At equilibrium, the probability of a neuron being in the state, p_{ij}, is estimated by running the network for a fixed time. The network is again run without clamping the output units and the probability p'_{ij} for each neuron pairs is estimated. The change in each of the weights is given by $\Delta w_{ji} = -\eta\ (p_{ji} - p'_{ji})$. where η is the learning rate.

1.2. NEURAL NETWORKS FOR PATTERN CLASSIFICATION

The objective of any pattern classifier is to assign the input patterns to one of a finite number of classes. The conventional pattern classifier consists of two stages: The first stage computes the matching scores for each class and the second selects the class with the maximum score. The input patterns can be viewed as points in the multidimensional space defined by input feature measurements. If the input vector is of dimension *N*, and the number of classes is *M*, the pattern classification could be viewed as partitioning the *N* dimension space into *M* dimensional regions[78].

In brief, it can be shown that the change in the network weight connecting neuron *j* in the lower layer to neuron *i* in the higher layer, during the p^{th} pattern presentation, is given by[14]:

$$\Delta w_{ji} = \eta \delta_{pj} \, o_{pi} \tag{2.4}$$

where

$$\delta_{pi} = (d_j - o_j) \, f'_j \tag{2.5}$$

for the output layer neurons and

$$\delta_{pi} = (\sum_k \delta_{pk} w_{kj}) \, f'_j \tag{2.6}$$

for any hidden layer neuron.

1.1.2. The Boltzmann Machine:

The Boltzmann machine is a generalization of the Hopfield networks in which the neurons update their states according to a stochastic decision rule[32][33]. The neurons have two states, on or off and the probability that the unit *i* adopts the state *'1'*, regardless of its previous states given by:

$$P_i = \frac{1}{1 + e^{-\Delta E_i / T}} \tag{2.7}$$

where ΔE_i is the total input received by the unit *i* and *T* is the 'temperature'. The decision rule given by the above equation is valid only for a particle which has two energy states and it can be shown that the system of such particles would reach a 'thermal equilibrium'[31].

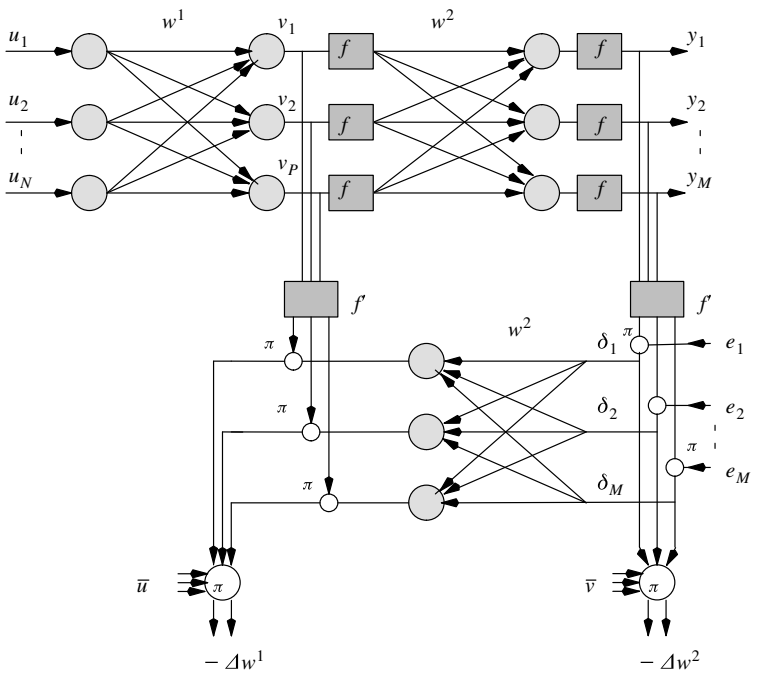
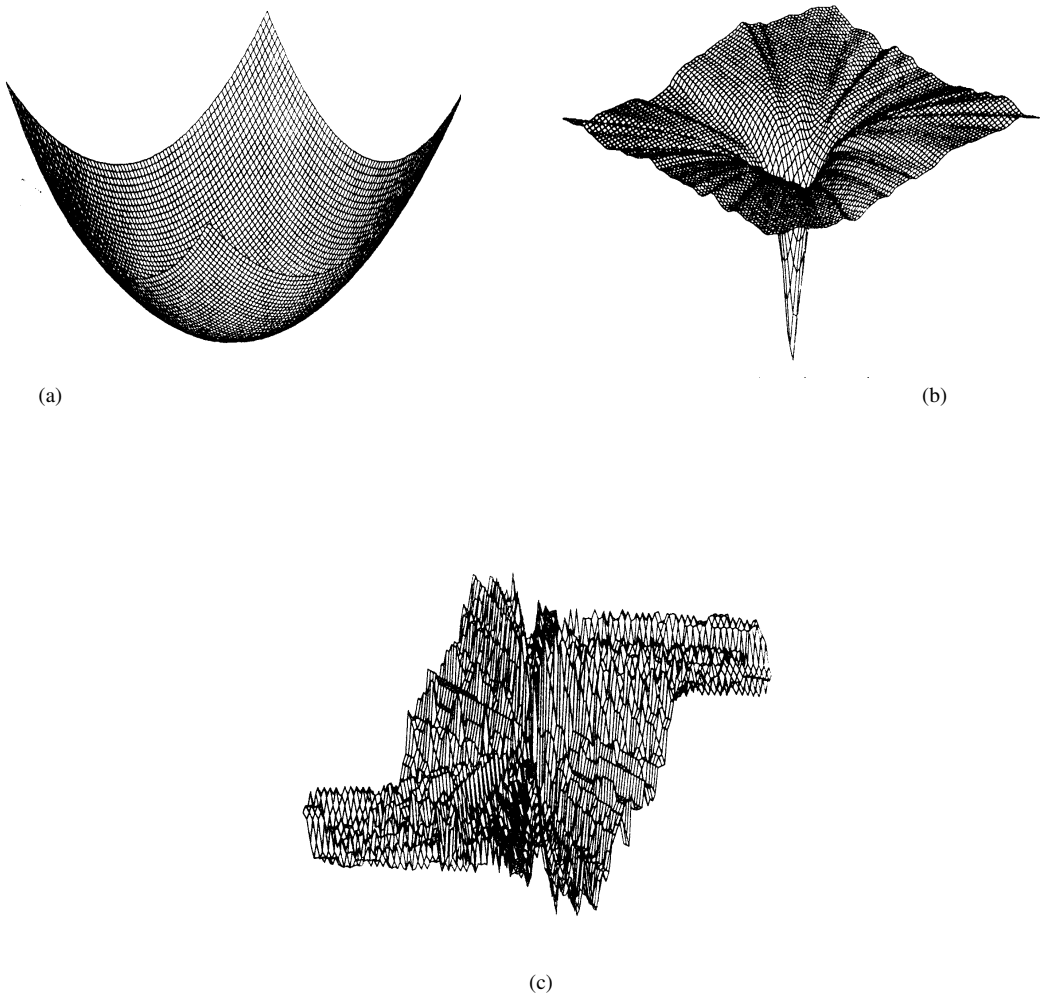


Figure 2.4 Diagrammatic Representation of Backpropagation (adopted from [77])

Figure 2.4 shows the diagrammatic representation of the backpropagation algorithm[77]. The inputs and outputs of the neural network are represented by $u^T = [u_1, u_2, ..., u_N]$, and $y^T = [y_1, y_2, ..., y_M]$ respectively. The nonlinear transformation function and its derivative are denoted by *f* and *f'*. The error derivatives for the output layer is backpropagated through the weights, as shown in the figure, to estimate the corresponding derivatives at the hidden layers. Note that the error propagating network is the transpose of the forward propagation network. The derivatives of the network parameters at any point in the network are available readily from the backpropagating part of the network. This property has some interesting use in the control applications and is further discussed in Chapter 6. A detailed explanation of the backpropagation algorithm is not given here. The basic principles of backpropagation, its modifications, limitations and applications, could be found in the literature [14][26][35] [75][77].

weights are estimated by propagating the error backwards from the output layer. Since this operation could be done only layer by layer, the weight adaptation is not synchronous.



(a) (b)

(c)

Figure 2.3 Mean Square Error surfaces for different transfer functions (adopted from [75])

(a) Linear Transfer Function
(b) Sigmoid Transfer Function
(c) Signum Transfer Function

$$E = \frac{1}{2}\sum_{j}(d_j - o_j^{(n)})^2 \tag{2.2}$$

where $o_j^{(n)}$ denote the actual state and d_j the desired output of the output neuron j.

We can minimize the error[1] E, by initializing the weights to random values and changing each weight by an amount proportional to the negative of the gradient of the error. That is,

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} \tag{2.3}$$

where η is a positive valued learning coefficient. Note that for a linear threshold transfer function, the mean squared error (MSE) surface is convex hyperparabolic, that never goes negative and has only one minimum. Hence the above given procedure guarantees convergence to the global error minimum. The error surface characteristics may vary according to the transfer function used at the neurons. For example, figure 2.3 shows the MSE surface of a single layer perceptron for linear, sigmoid and signum transfer functions[75]. It can be seen that the sigmoid and signum transformations distort the error surface considerably.

1.1.1. Multilayer Neural Networks: Backpropagation algorithm

The emergence of backpropagation algorithm is the most influential development in the field of neural networks in the past decade. The algorithm provides an effective way of training networks of multiple layers, there by enabling them to perform many of the complex tasks which were earlier thought as unsolvable by perceptrons. The backpropagation also uses the MSE measure given in equation (2.1). The gradient of error with respect to the hidden layer

---

1. Even though the most common error function is the mean squared error (MSE), Baum *et al.*[76] and Widrow *et al.*[75] point out that in certain occasions other error measures may be appropriate. We have observed that an information theoretic error measure[76] shows better convergence characteristics on a number of problems using the Alopex algorithm and the results are discussed in Chapter 3.

tion of the output for updating the weights. Figure 2.1 shows the taxonomy of some of the important learning algorithms for static pattern recognition. The binary input networks are the class of neural networks for which the inputs and outputs should be either '0' or '1', while the continuous input networks do not have such a restriction. As the dissertation deals only the supervised learning procedures, the discussion in this chapter is limited only to those concepts.

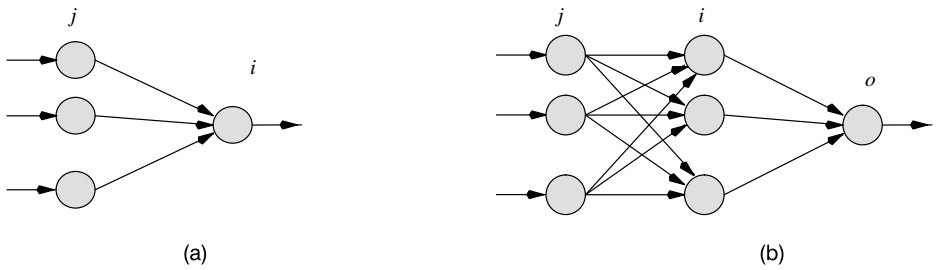


Fig. 2.2 Perceptrons

(a) Single Layer Architecture
(b) Multilayer Architecture

Consider a single layer perceptron with a nonlinear transformation function for the neurons as shown in figure 2.2a. The output of the neuron i at the n^{th} layer, is the summation of the contributions from the neurons in the $(n-1)^{\text{th}}$ layer, which is represented by:

$$o_i^{(n)} = f\left(\sum_j w_{ji} o_j^{(n-1)} + \theta_i\right) \tag{2.1}$$

where f denotes the nonlinear transformation function, normally a sigmoid function. The threshold for the neuron i is represented by θ_i. The performance of the network could be measured by the squared error function of the form,

rameter disturbances, measurement noise rejection and environmental disturbances[53][62]. Other comprehensively studied areas include signal processing and prediction[71], speech recognition[48][72] and fault detection[73]. This chapter reviews some of the important connectionist approaches used in pattern classification and control, which are used as the applications tools throughout this thesis. More rigorous review and analyses of some of the relevant approaches are given in the following chapters, as well.

1.1. SUPERVISED LEARNING ALGORITHMS

The learning algorithms are mainly divided into three categories, viz., supervised learning, unsupervised learning and the reinforcement procedures. The supervised learning procedures which require a teacher to specify the desired output vector; examples are backpropagation and Boltzmann machine. The unsupervised learning constructs internal models without any external supervision; ART, Kohonen's self organizing networks etc. are a few examples. The reinforcement procedures, such as A_{R-P} requires only a single scalar evalua–

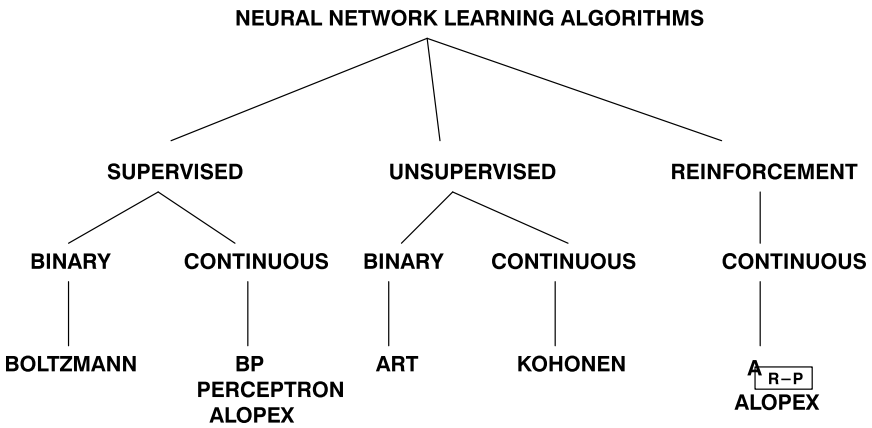


Figure 2.1 Categorization of different learning algorithms

CHAPTER 1.

LEARNING IN CONNECTIONIST MODELS

One of the important reasons for the increased interests in the recent times concerning the connectionist models, is the emergence of new network architectures, learning algorithms and the surmission that massive parallelism is essential for many of the information processing tasks which the conventional digital computers are unable to solve. Another reason is the technological advances in VLSI implementation and computer aided design schemes which allow the building of massively parallel computing machines[31].

The connectionist networks are found to show promising results in a number of practical problems. As classifiers, they provide reduced error rates compared to the traditional Bayesian approaches[68–70] and provide trade–offs in memory, computation, training time and adaptation requirements[46]. In the field of control, they offer a new approach for the identification and control of linear as well as nonlinear dynamical systems[26][51]. For a number of practical problems, they have demonstrated robustness and offer adaptability towards pa-