Invited Paper

Neural networks in engineering

D.T. Pham

Intelligent Systems Laboratory, School of Electrical, Electronic and Systems Engineering, University of Wales College of Cardiff, Cardiff, CF2 1XH, UK

Abstract

This paper divides neural networks into categories based on their structures and training methods and describes examples in each category. The paper outlines broad groups of engineering applications of neural networks, cites different applications in the major engineering disciplines and presents some recent applications investigated in the author's laboratory.

Introduction

Neural networks are computational models of the brain. There are over 50 different neural network models, some based more closely on current understanding of the brain operation than others. However, in general neural networks all have two of the brain's important characteristics: a parallel and distributed architecture and an ability to learn.

This paper aims to give a bird's eye view of the technology of neural computing and its engineering applications. The paper initially divides existing neural networks into categories according to their structures and learning methods and considers specific examples in each category. It then outlines broad groups of engineering applications of neural networks, cites different applications in the major engineering disciplines and presents some recent applications investigated in the author's laboratory.
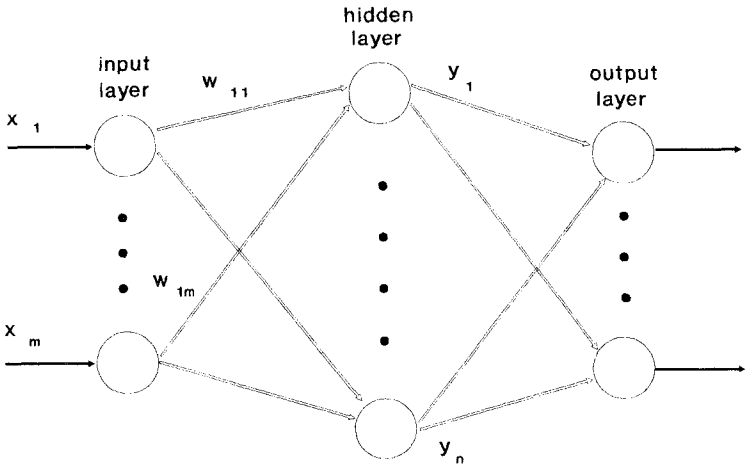
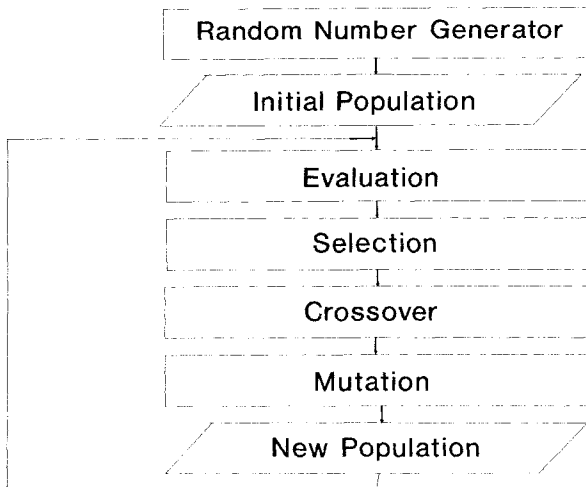Figure 1   A Multi-layer Perceptron

Figure 2 A simple genetic algorithm

where η is a parameter called the learning rate and δ_j is a factor depending on whether neuron j is an output neuron or a hidden neuron. For output neurons,

$$\delta_j = \left(\frac{df}{d\,net_j}\right)\left(y_j^{(t)} - y_j\right) \tag{3a}$$

and for hidden neurons,

$$\delta_j = \left(\frac{df}{d\,net_j}\right)\sum_q w_{qj}\,\delta_q \tag{3b}$$

In eqn (3(a)), net_j is the total weighted sum of input signals to neuron j and $y_j^{(t)}$ is the target output for neuron j.

As there are no target outputs for hidden neurons, in eqn (3(b)), the difference between the target and actual output of a hidden neuron j is replaced by the weighted sum of the δ_q terms already obtained for neurons q connected to the output of j.

Thus, iteratively, beginning with the output layer, the δ term is computed for neurons in all layers and weight updates determined for all connections.

Another learning algorithm suitable for training MLPs is the GA (see Figure 2). This is an optimisation algorithm based on evolution principles. The weights of the connections are considered genes in a chromosome. The goodness or fitness of the chromosome is directly related to how well trained the MLP is. The algorithm starts with a randomly generated population of chromosomes and applies genetic operators to create new and fitter populations. The most common genetic operators are the selection, crossover and mutation operators. The selection operator chooses chromosomes from the current population for reproduction. Usually, a biased selection procedure is adopted which favours the fitter chromosomes. The crossover operator creates two new chromosomes from two existing chromosomes by cutting them at a random position and exchanging the parts following the cut. The mutation operator produces a new chromosome by randomly changing the genes of an existing chromosome. Together, these operators simulate a guided random search method which can eventually yield the optimum set of weights to minimise the differences between the actual and target outputs of the neural network.

Learning vector quantization (LVQ) network

Figure 3 shows an LVQ network which comprises three layers of neurons: an input buffer layer, a hidden layer and an output layer. The network is fully connected between the input and hidden layers and partially connected between the hidden and output layers, with each output neuron linked to a different cluster of hidden neurons. The weights of the connections between the hidden and output neurons are fixed to 1. The weights of the input-hidden neuron
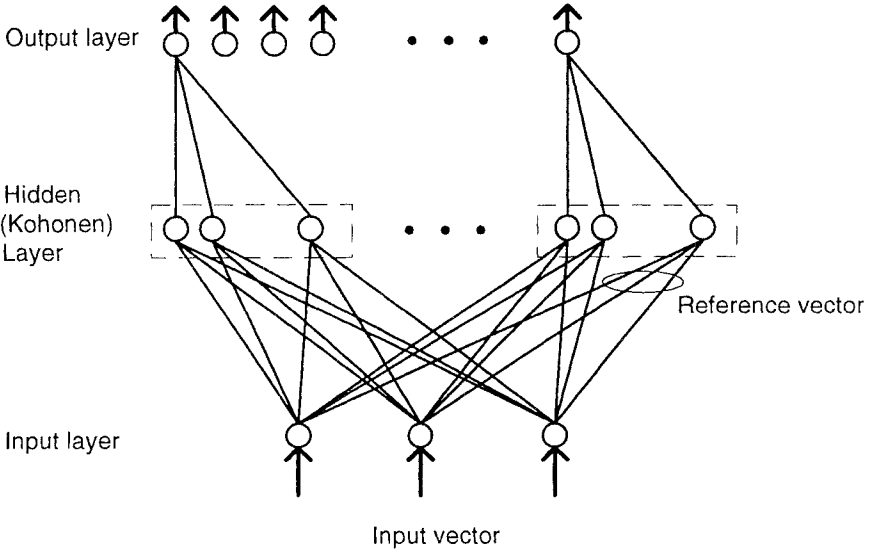
8   Artificial Intelligence in Engineering

Figure 3 Learning Vector Quantisation network

connections form the components of "reference" vectors (one reference vector is assigned to each hidden neuron). They are modified during the training of the network. Both the hidden neurons (also known as Kohonen neurons) and the output neurons have binary outputs. When an input pattern is supplied to the network, the hidden neuron whose reference vector is closest to the input pattern is said to win the competition for being activated and thus allowed to produce a "1". All other hidden neurons are forced to produce a "0". The output neuron connected to the cluster of hidden neurons that contains the winning neuron also emits a "1" and all other output neurons a "0". The output neuron that produces a "1" gives the class of the input pattern, each output neuron being dedicated to a different class. The simplest LVQ training procedure is as follows:-

(i)   initialise the weights of the reference vectors;
(ii)   present a training input pattern to the network;
(iii) calculate the (Euclidean) distance between the input pattern and each reference vector;
(iv)   update the weights of the reference vector that is closest to the input pattern, that is, the reference vector of the winning hidden neuron. If the latter belongs to the cluster connected to the output neuron in the class that the input pattern is known to belong to, the reference vector is brought closer to the input pattern. Otherwise, the reference vector is moved away from the input pattern;
(v)   return to (ii) with a new training input pattern and repeat the procedure until all training patterns are correctly classified (or a stopping criterion is met).

For other LVQ training procedures, see for example [Pham and Oztemel, 1994].

Group method of data handling (GMDH) network

Figure 4 shows a GMDH network and the details of one of its neurons. Unlike the feedforward neural networks previously described which have a fixed structure, a GMDH network has a structure which grows during training. Each neuron in a GMDH network usually has two inputs x_1 and x_2 and produces an output y that is a quadratic combination of these inputs, viz.

$$y = w_o + w_1 x_1 + w_2 x_1^2 + w_3 x_1 x_2 + w_4 x_2^2 + w_5 x_2 \tag{4}$$

Training a GMDH network consists of configuring the network starting with the input layer, adjusting the weights of each neuron, and increasing the number of layers until the accuracy of the mapping achieved with the network deteriorates.

The number of neurons in the first layer depends on the number of external inputs available. For each pair of external inputs, one neuron is used.
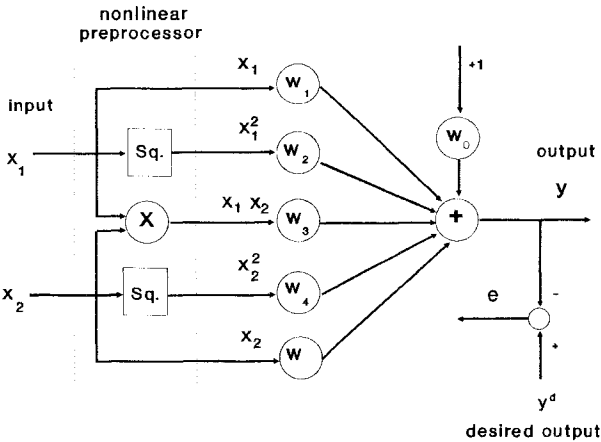
10   Artificial Intelligence in Engineering

Note: Each GMDH neuron is an N-Adaline, which is an Adaptive Linear Element
with a nonlinear preprocessor
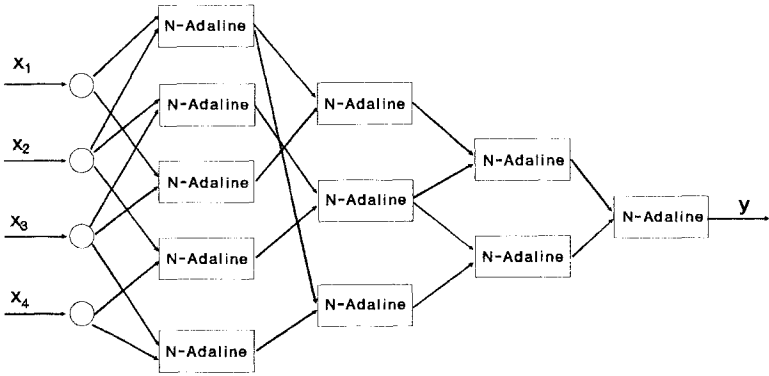
(a) A trained GMDH network

(b) Details of A GMDH Neuron

Figure 4 A GMDH network after training and
details of a GMDH neuron

Training proceeds with presenting an input pattern to the input layer and adapting the weights of each neuron according to a suitable learning algorithm, such as the delta rule (see for example [Pham and Liu, 1994]), viz.

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \alpha \frac{\mathbf{X}_k}{|\mathbf{X}_k|^2} \left(y_k^d - \mathbf{W}_k^T \mathbf{X}_k \right) \tag{5}$$

where $\mathbf{W}_k$, the weight vector of a neuron at time k, and $\mathbf{X}_k$ the modified input vector to the neuron at time k, are defined as

$$\mathbf{W}_k = [w_o \ \ w_1 \ w_2 \ \ w_3 \ w_4 \ \ w_5]^T \tag{6}$$

$$\mathbf{X}_k = [1 \ \ x_1 \ \ x_1^2 \ \ x_1 x_2 \ \ x_2^2 \ x_2]^T \tag{7}$$

and y_k^d is the desired network output at time k.

Note that, for this description, it is assumed that the GMDH network only has one output. Eqn (5) shows that the desired network output is presented to each neuron in the input layer and an attempt is made to train each neuron to produce that output. When the sum of the mean square errors S_E over all the desired outputs in the training data set for a given neuron reaches the minimum for that neuron, the weights of the neuron are frozen and its training halted. When the training has ended for all neurons in a layer, the training for the layer stops. Neurons that produce S_E values below a given threshold when another set of data (known as the selection data set) is presented to the network are selected to grow the next layer. At each stage, the smallest S_E value achieved for the selection data set is recorded. If the smallest S_E value for the current layer is less than that for the previous layer (that is, the accuracy of the network is improving), a new layer is generated, the size of which depends on the number of neurons just selected. The training and selection processes are repeated until the S_E value deteriorates. The best neuron in the immediately preceding layer is then taken as the output neuron for the network.

Hopfield net

Figure 5 shows one version of a Hopfield network. This network normally accepts binary and bipolar inputs (+1 or -1). It has a single "layer" of neurons, each connected to all the others, giving it a recurrent structure, as mentioned earlier. The "training" of a Hopfield network takes only one step, the weights w_{ij} of the network being assigned directly as follows:-

$$\begin{aligned} w_{ij} &= \frac{1}{N} \sum_{c=1}^{P} x_i^c \ x_j^c \quad i \neq j \\ &= \ \ 0 \qquad\qquad i = j \end{aligned} \tag{8}$$
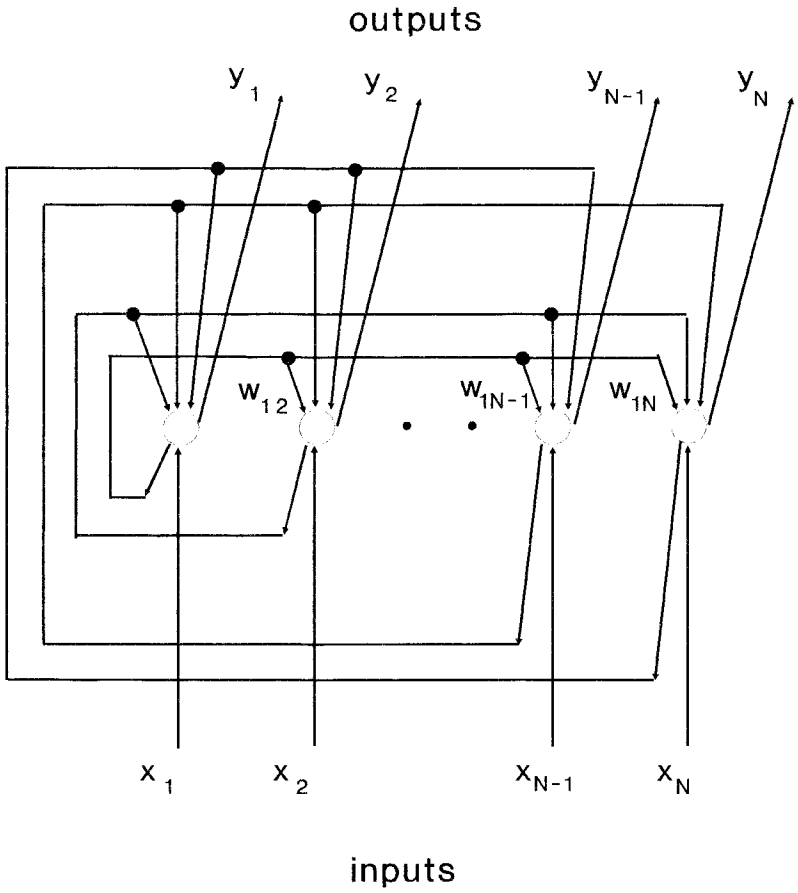
Figure 5 A Hopfield Network

where w_{ij} is the connection weight from neuron i to neuron j, and x_i^c (which is either +1 or -1) is the ith component of the training input pattern for class c, P the number of classes and N the number of neurons (or the number of components in the input pattern). Note from eqn (8) that $w_{ij} = w_{ji}$ and $w_{ii} = 0$, a set of conditions that guarantee the stability of the network. When an unknown pattern is input to the network, its outputs are initially set equal to the components of the unknown pattern, viz.

$$y_i(0) = x_i \qquad 1 \leq i \leq N \qquad\qquad (9)$$

Starting with these initial values, the network iterates according to the following equation until it reaches a minimum "energy" state, i.e. its outputs stabilise to constant values:-

$$y_i(k+1) = f\left[\sum_{j=1}^{N} w_{ij} \, y_i(k) \right] \qquad 1 < i \leq N \qquad\qquad (10)$$

where f is a hard limiting function defined as

$$\begin{aligned} f(x) &= -1 \quad x < 0 \\ &= 1 \quad\; x > 0 \end{aligned} \qquad\qquad (11)$$

Elman and Jordan nets

Figures 6(a) and (b) show an Elman net and a Jordan net, respectively. These networks have a multi-layered structure similar to the structure of MLPs. In both nets, in addition to an ordinary hidden layer, there is another special hidden layer sometimes called the context or state layer. This layer receives feedback signals from the ordinary hidden layer (in the case of an Elman net) or from the output layer (in the case of a Jordan net). The Jordan net also has connections from each neuron in the context layer back to itself. With both nets, the outputs of neurons in the context layer, are fed forward to the hidden layer. If only the forward connections are to be adapted and the feedback connections are preset to constant values, these networks can be considered ordinary feedforward networks and the BP algorithm used to train them. Otherwise, a GA could be employed [Pham and Karaboga, 1993]. For improved versions of the Elman and Jordan nets see [Pham and Liu, 1992; Pham and Oh, 1992].
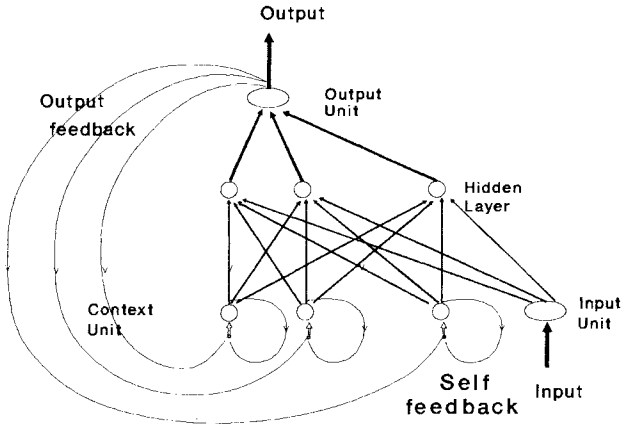
Kohonen network

A Kohonen network or a self-organising feature map has two layers, an input buffer layer to receive the input pattern and an output layer (see figure 7). Neurons in the output layer are usually arranged into a regular two-dimensional array. Each output neuron is connected to all input neurons. The weights of the

14   Artificial Intelligence in Engineering
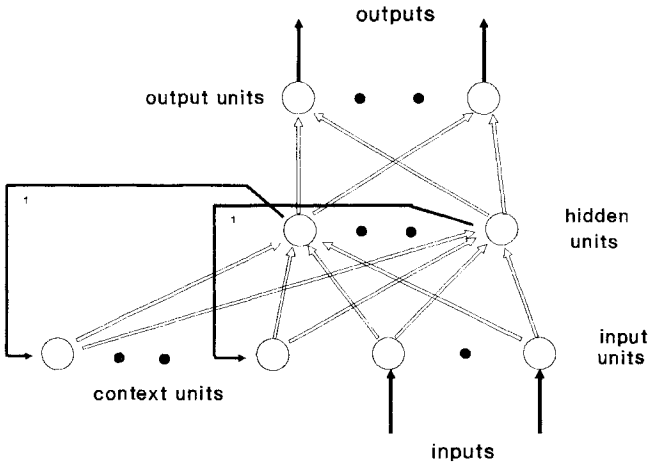
(a) An Elman Network

(b) A Jordan Network

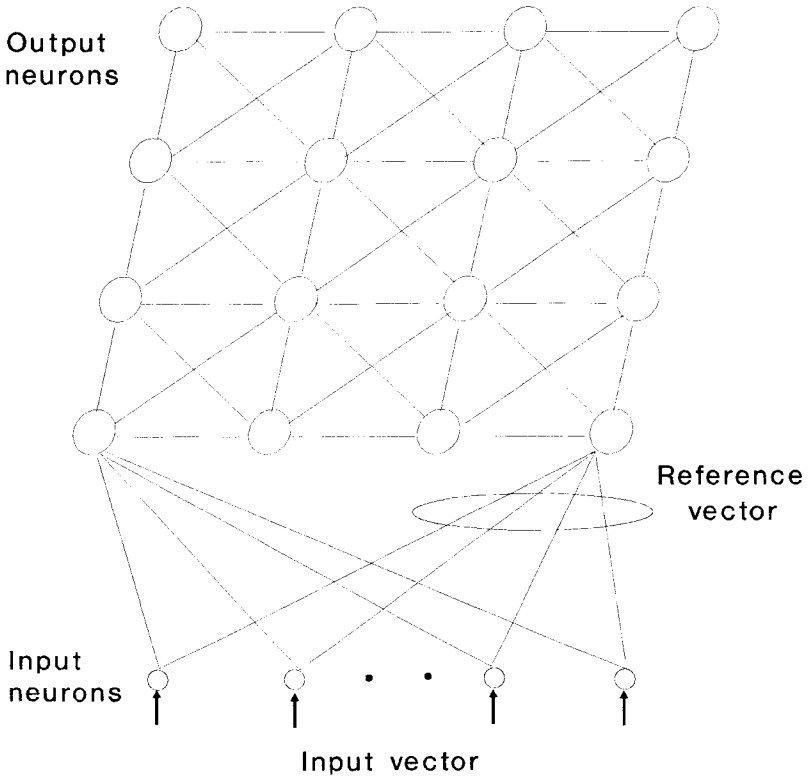Figure 6 An Elman network and
a Jordan network

Figure 7   A Kohonen network

16   Artificial Intelligence in Engineering

connections form the components of the reference vector associated with the given output neuron.

Training a Kohonen network involves the following steps:

(i)   initialise the reference vectors of all output neurons to small random values;

(ii) present a training input pattern;

(iii) determine the winning output neuron, i.e. the neuron whose reference vector is closest to the input pattern. The Euclidean distance between a reference vector and the input vector is usually adopted as the distance measure;

(iv) update the reference vector of the winning neuron and those of its neighbours. These reference vectors are brought closer to the input vector. The adjustment is greatest for the reference vector of the winning neuron and decreased for reference vectors of neurons further away. The size of the neighbourhood of a neuron is reduced as training proceeds until, towards the end of training, only the reference vector of a winning neuron is adjusted.

In a well-trained Kohonen network, output neurons that are close to one another have similar reference vectors. After training, a labelling procedure is adopted where input patterns of known classes are fed to the network and class labels are assigned to output neurons that are activated by those input patterns. As with the LVQ network, an output neuron is activated by an input pattern if it wins the competition against other output neurons, that is, if its reference vector is closest to the input pattern.

ART network

There are different versions of the ART network. Figure 8 shows the ART-1 version for dealing with binary inputs. Later versions, such as ART-2, can also handle continuous-valued inputs.

As illustrated in Figure 8, an ART-1 network has two layers, an input layer and an output layer. The two layers are fully interconnected, the connections are in both the forward (or bottom-up) direction and the feedback (or top-down) direction. The vector $\mathbf{W}_i$ of weights of the bottom-up connections to an output neuron i forms an exemplar of the class it represents. All the $\mathbf{W}_i$ vectors constitute the long-term memory of the network. They are employed to select the winning neuron, the latter again being the neuron whose $\mathbf{W}_i$ vector is most similar to the current input pattern. The vector $\mathbf{V}_i$ of the weights of the top-down connections from an output neuron i is used for "vigilance" testing, that is, determining whether an input pattern is sufficiently close to a stored exemplar. The vigilance vectors $\mathbf{V}_i$ form the short-term memory of the network. $\mathbf{V}_i$ and $\mathbf{W}_i$ are related in that $\mathbf{W}_i$ is a normalised copy of $\mathbf{V}_i$, viz.
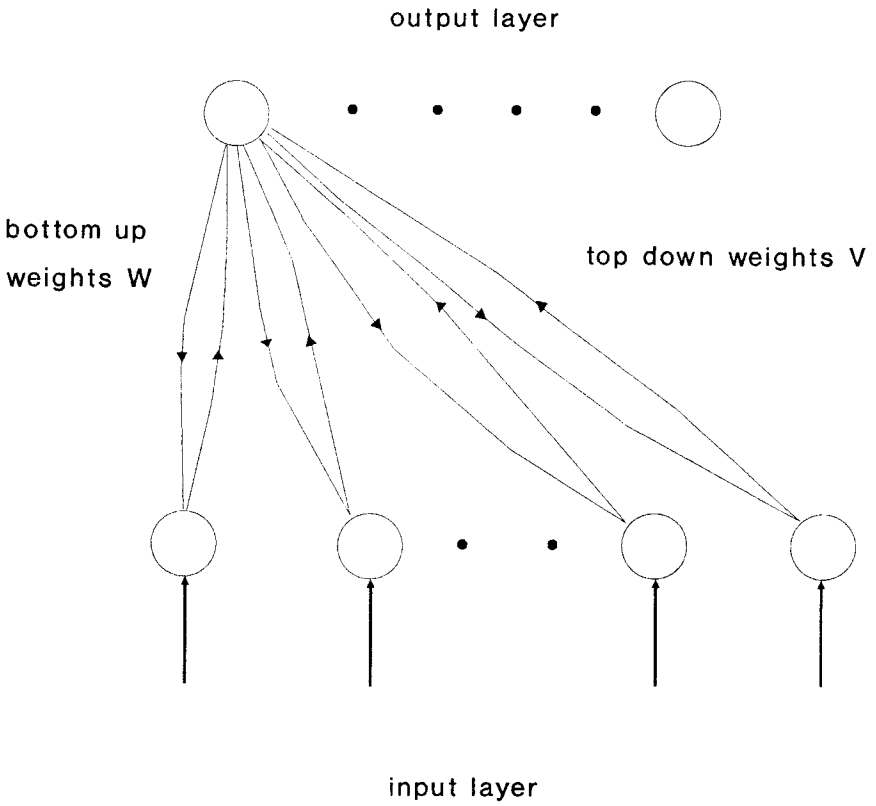
Figure 8   An ART1 Network

18   Artificial Intelligence in Engineering

$$\mathbf{W}_i = \frac{\mathbf{V}_i}{\varepsilon + \Sigma\, V_{ji}} \tag{12}$$

where ε is a small constant and V_{ji}, the jth component of $\mathbf{V}_i$ (i.e. the weight of the connection from output neuron i to input neuron j).

Training an ART-1 network occurs continuously when the network is in use and involves the following steps:-

(i)   initialise the exemplar and vigilance vectors $\mathbf{W}_i$ and $\mathbf{V}_i$ for all output neurons, setting all the components of each $\mathbf{V}_i$ to 1 and computing $\mathbf{W}_i$ according to eqn (12). An output neuron with all its vigilance weights set to 1 is known as an "uncommitted" neuron in the sense that it is not assigned to represent any pattern classes;

(ii) present a new input pattern $\mathbf{x}$;

(iii) enable all output neurons so that they can participate in the competition for activation;

(iv) find the winning output neuron among the competing neurons, i.e. the neuron for which $\mathbf{x}.\mathbf{W}_i$ is largest;   a winning neuron can be an uncommitted neuron as is the case at the beginning of training or if there are no better output neurons;

(v)   test whether the input pattern $\mathbf{x}$ is sufficiently similar to the vigilance vector $\mathbf{V}_i$ of the winning neuron. Similarity is measured by the fraction r of bits in $\mathbf{x}$ that are also in $\mathbf{V}_i$, viz.

$$r = \frac{\mathbf{x}.\mathbf{V}_i}{\Sigma x_i} \tag{13}$$

$\mathbf{x}$ is deemed to be sufficiently similar to $\mathbf{V}_i$ if r is at least equal to "vigilance threshold" ρ $(0 < \rho \le 1)$;

(vi) go to step (vii) if $r \ge \rho$ (i.e. there is "resonance"); else disable the winning neuron temporarily from further competition and go to step (iv) repeating this procedure until there are no further enabled neurons;

(vii) adjust the vigilance vector $\mathbf{V}_i$ of the most recent winning neuron by logically ANDing it with $\mathbf{x}$, thus deleting bits in $\mathbf{V}_i$ that are not also in $\mathbf{x}$; compute the bottom-up exemplar vector $\mathbf{W}_i$ using the new $\mathbf{V}_i$ according to eqn (12); activate the winning output neuron;

(viii) go to step (ii).

The above training procedure ensures that if the same sequence of training patterns is repeatedly presented to the network, its long-term and short-term memories are unchanged (i.e. the network is "stable"). Also, provided there are sufficient output neurons to represent all the different classes, new patterns can always be learnt, as a new pattern can be assigned to an uncommitted output

neuron if it does not match previously stored exemplars well (i.e. the network is "plastic").

Neural network applications

Types of applications

Neural networks have been employed in a wide range of applications. These can be grouped into six general types which are [NeuralWare, 1993]:-

Modelling and prediction   A modelling neural network essentially acts as a mapping operator or transfer function. It models a process or system, taking inputs which are normally fed to that process or system and computing its predicted output values. For example, a neural network modelling an electric furnace might receive values of the input currents supplied to the furnace and produce the corresponding furnace temperature values. The input-output mapping can be static or dynamic. The most commonly used neural network for static mapping applications is the MLP. MLPs, GMDH networks and recurrent networks such as the Elman and Jordan nets have been employed for dynamic mapping applications.

Classification and pattern recognition   A classification neural network receives input patterns describing an object or situation and outputs the category of that object or situation. In the case of a fault diagnosis application, for instance, input patterns might be symptoms and test data and the network output, the fault type. Again MLPs have been used for classification tasks. Another frequently adopted classification network is the LVQ network.

Association   Neural networks can be employed as pattern associators and auto-associative content-addressable memories. An associative network is taught associations of ideal noise-free data and subsequently can classify data that is corrupted by noise. For example, the network learns a number of ideal patterns and recognises a noisy input pattern as one of the taught ideal patterns. This recognition process can involve completing a corrupted input pattern by replacing its missing parts. Historically, a well-known associative network is the Hopfield net although the related Hamming net [Lippmann, 1987] is a faster and more accurate network with a higher storage capacity. An MLP can also be used as an auto-associative network by arranging for the input and output layers to have the same number of neurons and making the target output the same as the input for all training examples.

Clustering   This type of application is normally implemented with unsupervised neural networks such as the ART network and Kohonen's self-organising feature map. Input vectors representing the attributes or features of different objects or situations are supplied to a clustering network to analyse.

Based on correlations in the input vectors, the network groups them into categories. For example, input vectors can encode the geometrical features and machining requirements of workpieces and the network arranges them into group technology families.

Signal processing This class of applications includes data compression and noise filtering, both of which can be realised using MLPs in the auto-associative mode mentioned previously. Compression is achieved by adopting fewer hidden neurons than input and output neurons. Smoothing of a signal is implemented by first compressing and then reconstructing it.

Optimisation This can be carried out using a neural network designed inherently to minimize an energy function, such as the Hopfield net. The objective function and constraints of an optimisation problem are coded into the energy function of the network which aims to reach a stable state where its outputs yield the desired optimum parameters. Another optimisation approach can involve first training a neural model of the process to be optimised, using a suitable network such as an MLP, and then adapting the inputs to the model to produce the optimum process output. The input adaption procedure can be the same as that for weight adaptation during the training of the neural model.

Engineering applications
Applications of neural networks have been developed for problems in all major engineering disciplines. The following is a non-exhaustive list of applications in chemical and process engineering, civil and structural engineering, electrical and electronic engineering, manufacturing and mechanical engineering and systems and control engineering.

Chemical and process engineering
- predicting the colour of a product such as paint from the concentrations of the colorants used [Bishop et al, 1991];
- selecting chemical reactors [Bulsari and Saxen, 1992];
- diagnosing faults in dynamic processes [Vaidyanathan and Venkatasubramanian, 1992];
- estimating the melt flow index in industrial polymerisation [Peel et al, 1993];
- modelling an industrial fermentation process [Tsaptsinos et al, 1993];
- designing distillation columns [Mogili and Sunol, 1993];
- estimating microbial concentrations in a biochemical process [Bulsari, 1994].

Civil and structural engineering
- resource levelling in PERT analysis for construction projects [Shimazaki et al, 1991];
- controlling the discharge from a reservoir [Sakakima et al, 1992];

- classifying river water quality using biological data [Ruck et al, 1993];
- estimating the velocity, axle spacings and axle loads of a passing truck from strain response readings taken from the bridge it passes over [Gagarin et al, 1994];
- predicting the flow of a river from historical river flow data [Karunanithi et al, 1994];
- modelling a finite-element-based structural analysis procedure [Rogers, 1994];
- detecting flaws in the internal structure of construction components [Flood and Kartam, 1994].

Electrical and electronic engineering
- monitoring the alarm state of critically ill patients [Dodd, 1991];
- enhancing noisy images [Shih et al, 1992]
- compressing images [Lu and Shin, 1992];
- filtering noise and extracting edges from images [Pham and Bayro-Corrochano, 1992];
- detecting features in pulsed-radar backscatter waveforms [Mesher and Poehlman, 1992];
- estimating harmonic components in a power system [Osowski, 1992];
- forecasting daily peak loads in power systems [Ishibashi et al, 1992];
- allocating channels in mobile communication systems [Fritsch et al, 1993].
- classifying objects from their ultrasonic signatures [Smith et al, 1993];
- preprocessing images for an optical character recognition system [Holeva and Kadaba, 1993];

Manufacturing and mechanical engineering
- controlling the feedrate of a metal cutting machine tool [Epstein and Wright, 1991];
- monitoring tool wear by sensing acoustic emission and tool forces [Burke and Rangwala, 1991] and power consumption and tool acceleration [Wu, 1991];
- designing group-technology part families for cellular manufacturing [Malave and Ramachandran, 1991];
- planning collision-free paths for moving objects [Lee and Park, 1991];
- predicting chip breakability and surface finish in metal machining [Yao and Fang, 1993];
- optimising machining parameters [Wang, 1993];
- classifying machine faults [Huang and Wang, 1993];
- determining the external heat transfer across a one-dimensional body from the temperature history at internal points [Dumek et al, 1993];
- designing the configuration of aircraft wing box structures [Wang et al, 1993].

Systems and control engineering
- modelling the forward and inverse dynamics of plants and processes [Pham and Liu, 1992; Pham and Oh, 1992; Pham and Oh, 1993];
- controlling a flexible robot arm [Li et al, 1992];
- controlling the hovering of a model helicopter [Pallett and Ahmad, 1992];
- coordinating the trajectories of multiple robots [Srinivasan et al, 1992];
- controlling fluid levels in a two-tank system [Evans et al, 1993];
- controlling the batch quality in a distillation process [Cressy et al, 1993];
- measuring and controlling the depth of anaesthesia [Linkens and Rehman, 1993];
- determining an optimal path for a robot [Houillon and Caron, 1993];
- controlling the temperature of a water bath [Khalid et al, 1994].

Example applications
This section describes three applications recently investigated in the author's laboratory.

Control chart pattern recognition   A control chart for monitoring process parameters can exhibit different types of patterns. The six most common types are: normal patterns with random variations of a process parameter about a steady mean value, increasing or decreasing trends, increasing or decreasing shifts and cyclic patterns. It is useful to be able to recognise these patterns as they can indicate the long-term health of a process. Before neural networks were used, other methods of pattern recognition had been employed, including methods based on statistical analysis and expert system rules [Pham and Oztemel, 1992a]. However, the detection accuracy achieved had not been very high for patterns not clearly belonging to one of the known types. Figure 9 shows a three-layer MLP configured to recognise patterns that are time-series of 60 consecutive samples. The MLP has 60 input neurons (one for each sample), 35 hidden neurons and six output neurons (one for each pattern type). The MLP was trained on a set of 498 known patterns and tested on 1002 previously unseen patterns. Its best detection accuracy, achieved after 400 sequential presentations of the complete training set, was 96% [Pham and Oztemel, 1992b].

Figure 10 shows a system comprising three MLP pattern recognition modules coordinated by a rule-based module. Due to "synergy" between the modules, the best detection accuracy increased to 97.1% [Pham and Oztemel, 1992b]. Subsequent work has achieved even greater accuracies through combining different types of pattern recognition modules, for example, rule-based and MLP modules [Pham and Oztemel, 1993].

Automotive valve stem seal inspection   As critical components in a car engine, valve stem seals require 100% inspection. It is desirable to automate the inspection process because of the high production volume involved and the difficulty of achieving consistent results with human operators. A neural-
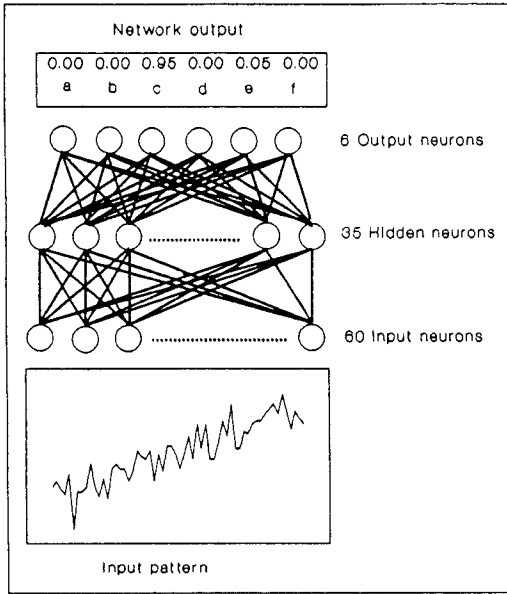
Figure 9   A control chart pattern recognition
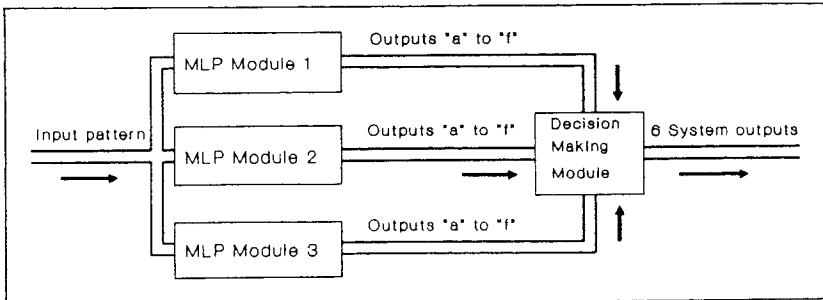module based on an MLP

Figure 10   A composite pattern recognition
system

network based system for automated visual inspection of up to 10 million seals per year is under development. The hardware of the system, depicted in Figure 11, comprises four CCD cameras linked to a vision computer. The cameras capture different views of a seal. Each camera image is processed using a conventional algorithm (histogram-based thresholding, connectivity analysis, labelling and Laplacian edge detection) to obtain a binary outline image. A two-layer perceptron with a competitive (MAXNET) layer and an accumulative layer is then employed to extract features of objects in the outline image (Figure 12a). The function of the neural feature extractor is to produce a signature for each object. This signature is a vector of 20 components each giving the number of times a particular geometric feature is present on the object contour (Figure 12b). Two other neural networks are employed to handle two inspection tasks, perimeter inspection and surface inspection. The perimeter inspection neural network is a three-layer MLP with 20 input neurons (for the 20 components of the signature vector), 10 hidden neurons and three output neurons (for the three possible types of seal perimeters). The surface inspection neural network is a three-layer MLP with 25 input neurons (for the 20 components of the signature vector plus five additional geometric parameters), 10 hidden neurons and three output neurons (for the three possible types of seal surface flaws). Preliminary tests have yielded a detection accuracy of 83% and 93% in the perimeter and surface inspection tasks respectively [Pham and Bayro-Corrochano, 1994].

Adaptive control of a robot arm   A robot arm is a multi-input-multi-output time-variant dynamic system. Effective control of a robot arm requires a multi-variable controller capable of adapting to unpredictable changes in the robot dynamics. Figure 13(a) shows such an adaptive controller for a two-joint SCARA robot [Pham and Oh, 1994 a and b]. The controller incorporates three neural networks for on-line direct learning of the robot forward and inverse dynamics. All three neural networks are based on the modified Jordan network described in [Pham and Oh, 1992] (see Figure 13(b)). The first neural network (ψ) learns to model the forward dynamics of the robot. ψ has two input neurons (one for each of the joint torques), two output neurons (one for each of the joint actuator rotation angles) and twelve hidden and context neurons. The second neural network (ϕ) learns to model the inverse dynamics of the robot. ϕ has four input neurons (two for the joint rotation angles and two for errors in rotation angles), two output neurons (for the two joint torques) and ten hidden and context neurons. The third neural network (ϕ^c) acts as a feedforward controller. ϕ^c is a copy of ϕ. The decision as to when to update ϕ^c with the most recent version of ϕ is made by an adaptation critic. A conventional feedback controller is also employed to compensate for imperfections in the inverse dynamics model. The design of the feedback controller can be accomplished almost arbitrarily, unlike the case when the neural controller is not in use. Figure 14 shows simulation results which demonstrate the control system successfully handling a sudden change in its payload.
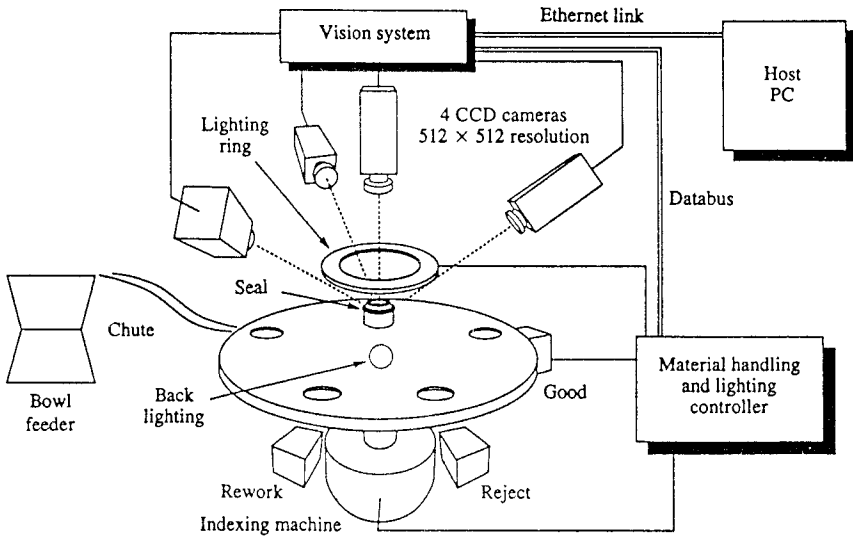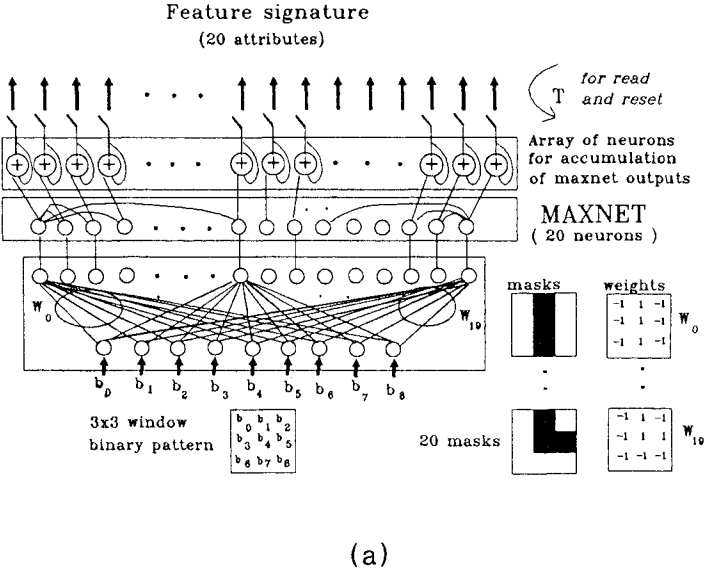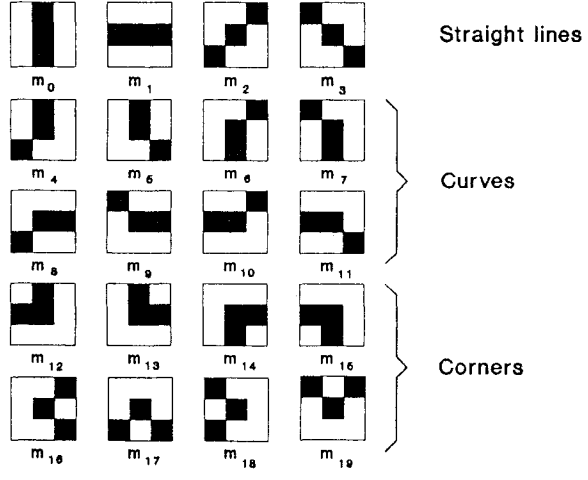
Figure 11   Automated visual inspection system

Feature signature
(20 attributes)

(a)

Straight lines

Curves

Corners

(b)

Figure 12 (a) Feature extraction neural network
(b) Elementary geometric features to
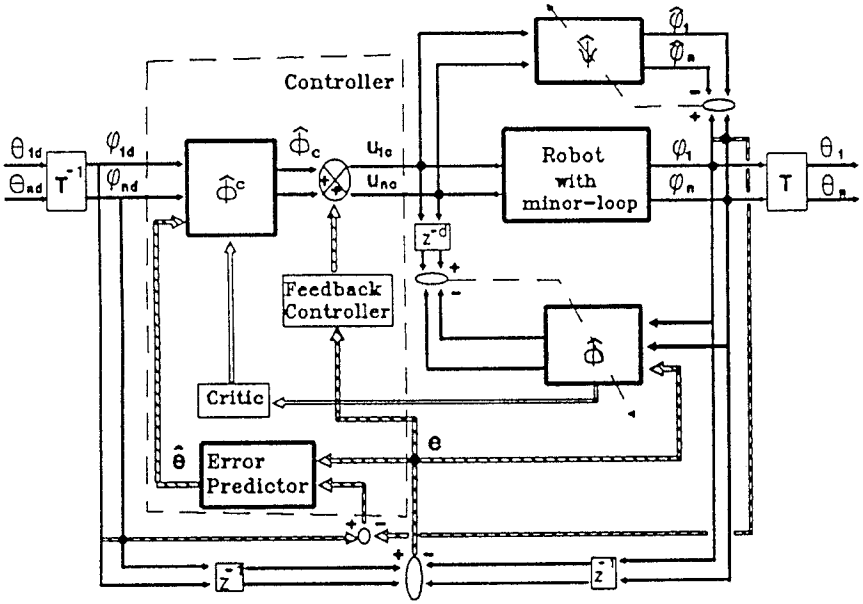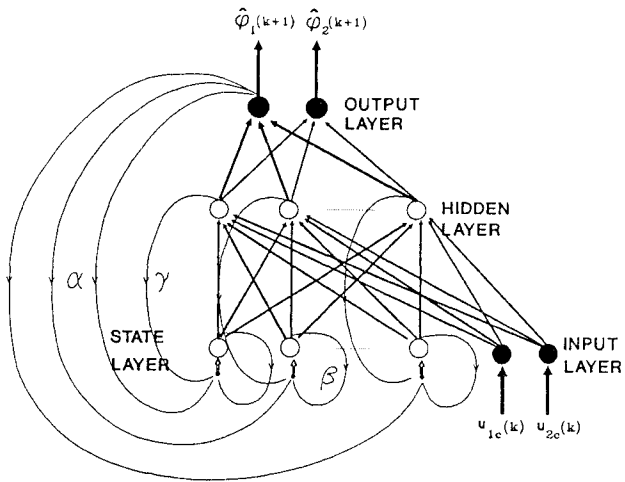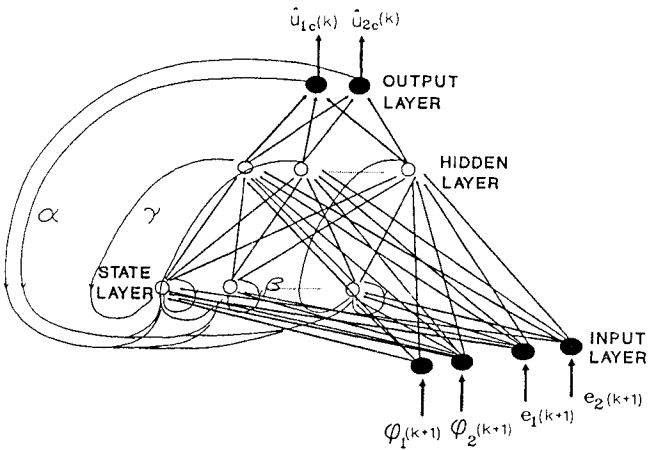be detected

Figure 13   (a) Control system for a two-joint
SCARA robot

(i)

(ii)

Figure 13   (b) Modified Jordan networks for (i)
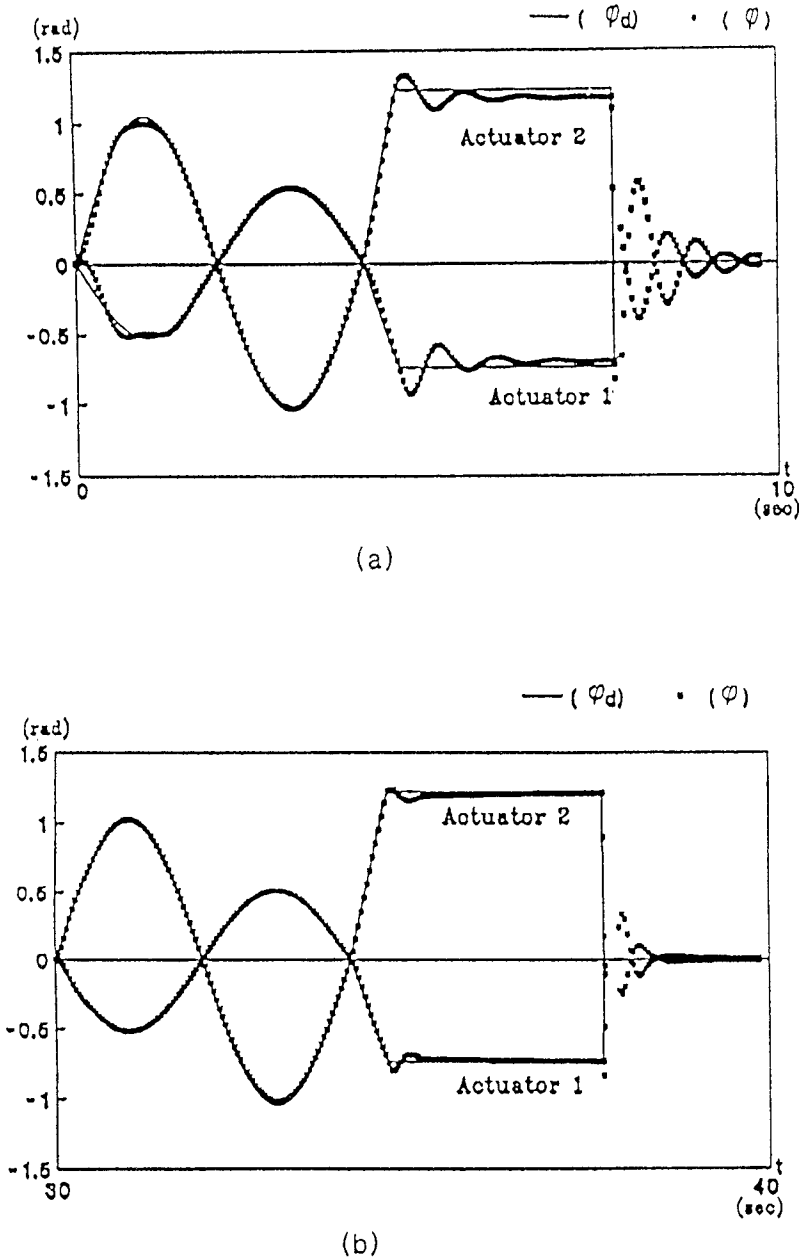forward and (ii) inverse dynamics
modelling

(a)

(b)

Figure 14   (a) Response of the robot joint actuators immediately after a load change
(b) Response after adaptation to the load change

Conclusion

Neural networks are computational systems that do not require programming in a conventional sense, but can learn and generalise from training examples. Neural computing is therefore potentially simple to apply and has generated a great deal of interest in engineering. Several applications of this technology have been proposed, although the majority of them are still in laboratory stages. A very large number of applications remain to be exploited and implemented in practice once neural computing has become more accessible to the wider audience of engineers in industry. In spite of the many different neural network paradigms in existence, as can be seen in this paper, they can be grouped into a small number of categories, which should facilitate familiarisation with the technology.

Finally, though powerful as it is, neural computing should not be regarded as a panacea for all problems. It is likely that future applications would benefit from judiciously combining neural computing and other AI technologies including expert systems, machine induction and fuzzy logic.

Acknowledgements

References

Bishop, J.M., Bushnell, M.J., Usher, A. and Westland, S. Neural networks in the colour industry. In: *Applications of Artificial Intelligence in Engineering VI*, (eds G. Rzevski and R.A. Adey), Computational Mechanics, Southampton, 423-434, 1991.

Bulsari, A.B. and Saxen, H. Implementation of a chemical reactor selection expert system in an artificial neural network, *Eng. App. of Artificial Intelligence,* 5(2), 113-119, 1992.

Bulsari, A.B. Applications of artificial neural networks in process engineering, *J. of Systems Engineering,* 4(2), 1994 (in press).

Burke, L.I. and Rangwala, S. Tool condition monitoring in metal cutting : a neural network approach, *J. of Intelligent Manufacturing,* 2(5), 269-279, 1991.

Carpenter, G.A. and Grossberg, S. The ART of adaptive pattern recognition by a self-organising neural network, *Computer,* March 1988, 77-88.

Cottrell, G.W., Munro, P. and Zipser, D. Learning internal representations from gray-scale images : an example of extensional programming, *Proc. 9th Annual Conf. of the Cognitive Science Society,* Seattle, 461-473, 1987.

Cressy, D.C., Nabney, I.T. and Simper, A.M. Neural control of a batch distillation, *Neural Computing and Applications,* 1(2), 115-123, 1993.

Dodd, N. Artificial neural network for alarm-state monitoring. In: *Artificial Intelligence in Engineering VI,* (eds G. Rzevski and R.A. Adey), Computational Mechanics, Southampton, 623-631, 1991.

Dumek, V., Druckmuller, M. and Raudensky, M. Neural network, expert system and inverse problems. In: *Applications of Artificial Intelligence in Engineering VIII,* Vol.1, (eds G. Rzevski, J. Pastor and R.A. Adey), Computational Mechanics, Southampton, 463-473, 1993.

Elman, J.L. Finding structure in time, *Cognitive Science,* 14, 179-211, 1990.

Epstein, H.A. and Wright, P.K. Intelligent machine tools : an application of neural networks to the control of cutting tool performance. In: A*rtificial Intelligence in Engineering VI,* (eds G. Rzevski and R.A. Adey), Computational Mechanics, Southampton, 597-609, 1991.

Evans, J.T., Gomm, J.B., Williams, D., Lisbon, P.J.G. and To, Q.S. A practical application of neural modelling and predictive control. In: *Application of Neural Networks to Modelling and Control,* (eds G.F. Page, J.B. Gomm and D. Williams), Chapman and Hall, London, 74-88, 1993.

Flood, I. and Kartam, N. Neural networks in civil engineering II : systems and application, *J. of Computing in Civil Engineering,* ASCE, 8(2), 149-162, 1994.

Fritsch, T., Mittler, M. and Tran-Gia, P. Artificial neural net applications in telecommunication systems, *Neural Computing and Applications,* 1(2), 124-146, 1993.

Gagarin, N., Flood, I. and Albrecht, P. Computing truck attributes with artificial neural networks, *J. of Computing in Civil Engineering,* ASCE, 8(2), 179-200, 1994.

Goldberg, D. *Genetic algorithms in search, optimisation and machine learning,* Addison-Wesley, Reading, MA, 1989.

Hecht-Nielsen, R. *Neurocomputing,* Addison-Wesley, Reading, MA, 1990.

Holeva, L.F. and Kadaba, N. An image filtering module using neural networks for an optical character recognising system, *Mathematical Modelling and Scientific Computing,* 2, Section A, 504-510, 1993.

Holland, J.H. *Adaptation in natural and artificial systems,* University of Michigan Press, Ann Arbor, MI, 1975.

Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities, *Proc. National Academy of Sciences,* USA, April, 79, 2554-2558, 1982.

Houillon, P. and Caron, A. Planar robot control in cluttered space by artificial neural network, *Mathematical Modelling and Scientific Computing,* 2, Section A, 498-503, 1993.

Huang, H.H. and Wang, H.P. Machine fault classification using an ART 2 neural network, *Int. J. Adv. Manuf. Technology,* 8(4), 194-199, 1993.

Ishibashi, K., Komura, T., Ueki, Y., Nakanishi, Y. and Matsui, T. Short-term load forecasting using an artificial neural network, *Proc. 2nd Int. Conf. on Automation, Robotics and Computer Vision,* Vol.3, September 1992, Singapore, INV-11.1.1-INV-11.1.5, 1992.

Jordan, M.I. Attractor dynamics and parallelism in a connectionist sequential machine, *Proc. 8th Annual Conf. of the Cognitive Science Society,* Amherst, MA, 531-546, 1986.

Karunanithi, N., Grenney, W.J., Whitley, D. and Bovee, K. Neural networks for river flow prediction, *J. of Computing in Civil Engineering,* ASCE, 8(2), 201-220, 1994.

Khalid, M., Omatu, S. and Yusof, R. Adaptive fuzzy control of a water bath process with neural networks, *Engineering Applications of Artificial Intelligence,* 7(1), 39-52, 1994.

Kohonen, T. *Self-organisation and associative memory* (3rd ed), Springer-Verlag, Berlin, 1989.

Lee, S. and Park, J. Neural computation for collision-free path planning, *J. of Intelligent Manufacturing* 2(5), 315-326.

Li, Y., Wong, A.K.C. and Yang, F. Neural network control of flexible robot arm. In: *Artificial Intelligence in Engineering VII,* (eds D.E. Grierson, G.

Rzevski and R.A. Adey), Computational Mechanics, Southampton, 89-105, 1992.

Linkens, D.A. and Rehman, H.U. Neural network controller for depth of anaesthesia. In: *Application of Neural Networks to Modelling and Control,* (eds G.F. Page, J.B. Gomm and D. Williams), Chapman and Hall, London, 104-115, 1993.

Lippmann, R.P. *An introduction to computing with neural nets, IEEE ASSP Magazine,* April, 4-22, 1987.

Lu, C.C. and Shin, Y.H. Neural networks for classified vector quantization of images, *Eng. Applications of Artificial Intelligence,* 5(5), 451-456, 1992.

Malave, C.O. and Ramachandran, S. Neural-network based design of cellular manufacturing systems, *J. of Intelligent Manufacturing,* 2(5), 305-314, 1991.

Mesher, D.E. and Poehlman, W.F.S. Interpretation of pulsed radar backscatter waveforms using a knowledge based system. In: *Artificial Intelligence in Engineering VII,* (eds D.E. Grierson, G. Rzevski and R.A. Adey), Computational Mechanics, Southampton, 255-269, 1992.

Mogili, P.K. and Sunol, A.K. Machine learning approach to design of complex distillation columns. In: *Applications of Artificial Intelligence in Engineering VIII,* Vol.2, (eds G. Rzevski, J. Pastor and R.A. Adey), Computational Mechanics, Southampton, 755-770, 1993.

NeuralWare Inc. *Neural Computing : NeuralWorks Professional II/PLUS and NeuralWorks Explorer,* NeuralWare Inc., Pittsburgh, Pennsylvania, 1993.

Pallett, T.J. and Ahmad, S. Real-time neural network control of a miniature helicopter in vertical flight. In: *Artificial Intelligence in Engineering VII,* (eds D.E. Grierson, G. Rzevski and R.A. Adey), Computational Mechanics, 143-160, 1992.

Peel, C., Willis, M.J. and Tham, M.T. Enhancing feedforward neural network training. In: *Application of Neural Networks to Modelling and Control,* (eds G.F. Page, J.B. Gomm and D. Williams), Chapman and Hall, London, 35-52, 1993.

Pham, D.T. and Bayro-Corrochano, E.J. Neural networks for noise filtering, edge detection and feature extraction, *J. of Systems Engineering,* 2(2), 111-122, 1992.

Pham, D.T. and Bayro-Corrochano, E.J. Neural classifiers for automated visual inspection, *IMechE Proc., Part D, J. of Automobile Engineering,* 208, 83-89, 1994.

Pham, D.T. and Karaboga, D. Dynamic system identification using recurrent neural networks and genetic algorithms, *Proc. 9th Int. Conf. on Mathematical and Computer Modelling,* San Francisco, July 1993 (in press).

Pham, D.T. and Liu, X. Dynamic system modelling using partially recurrent neural networks, *J. of Systems Engineering,* 2(2), 90-97, 1992.

Pham, D.T. and Liu, X. Modelling and production using GMDH networks of adalines with nonlinear preprocessors, *Int. J. of Systems Science,* (in press), 1994.

Pham, D.T. and Oh, S.J. A recurrent backpropagation neural network for dynamic system identification, *J. of Systems Engineering,* 2(4), 213-223, 1992.

Pham, D.T. and Oh, S.J. Identification of plant inverse dynamics using neural networks, Technical Report, Intelligent Systems Laboratory, School of Electrical, Electronic and Systems Engineering, University of Wales, Cardiff, UK. (1993).

Pham, D.T. and Oh, S.J. Adaptive control of a robot using neural networks, *Robotica,* 12, 1994a (in press).

Pham, D.T. and Oh, S.J. Control of a two-axis robot using neural networks, *Proc. IMACS Int. Symp. on Signal Processing, Robotics and Neural Networks,* Lille, April 1994b (in press).

Pham, D.T. and Oztemel, E. A knowledge-based statistical process control system, *Proc. 2nd Int. Conf. on Automation, Robotics and Computer Vision,* Vol.3, Singapore, Sept.16-18, INV-4.2.1-INV-4.2.6, 1992.

Pham, D.T. and Oztemel, E. Control chart pattern recognition using neural networks, *J. of Systems Engineering,* 2(4), 256-262, 1992.

Pham, D.T. and Oztemel, E. Combining multi-layer perceptrons with heuristics for reliable control chart pattern recognition. In: *Artificial Intelligence in Engineering VIII,* (eds G. Rzevski, J. Pastor and R.A. Adey), Computational Mechanics, Southampton, 801-810, 1993.

Pham, D.T. and Oztemel, E. Control chart pattern recognition using learning vector quantization networks, *Int. J. Prod. Research,* 32(3), 721-729, 1994.

Rogers, J.L.   Simulating structural analysis with neural network, *J. of Computing in Civil Engineering,* ASCE, 8(2), 252-265, 1994.

Ruck, B.M., Walley, W.J. and Hawkes, H.A. Biological classification of river water quality using neural networks. In: *Artificial Intelligence in Engineering VIII,* (eds G. Rzevski, J. Pastor and R.A. Adey), Computational Mechanics, Southampton, 361-372, 1993.

Rumelhart, D.E. and McClelland, J.L. *Parallel Distributed Processing : explorations in the microstructure of cognition,* MIT Press, Cambridge, MA, 1986.

Sakakima, S., Kojiri, T. and Itoh, K. Real-time reservoir operation with neural nets concept. In: *Artificial Intelligence in Engineering VII,* (eds D.E. Grierson, G. Rzevski and R.A. Adey), Computational Mechanics, Southampton, 501-514, 1992.

Shih, F.Y., Moh, J. and Bourne, H. A neural architecture applied to the enhancement of noisy binary images, *Eng. Applications of Artificial Intelligence,* 5(3), 215-222, 1992.

Shimazaki, T., Sand, K. and Tuchiya, Y. Resource levelling in PERT by neural network. In: *Artificial Intelligence in Engineering VI,* (eds G. Rzevski and R.A. Adey), Computational Mechanics, Southampton, 487-498, 1991.

Smith, P.D., Bull, D.R. and Wykes, C. Target classification with artificial neural networks using ultrasonic phased arrays. In: *Artificial Intelligence in Engineering VIII,* Vol.2, (eds G. Rzevski, J. Pastor and R.A. Adey, Computational Mechanics, Southampton, 787-800, 1993.

Srinivasan, S., Gander, R.E. and Wood, H.C. An artificial neural network for trajectory generation and control in a multiple robot system, *Proc. 2nd Int. Conf. on Automation, Robotics and Computer Vision,* Vol.3, September 1992, Singapore, RO-5.4.1-RO-5.4.5, 1992.

Tsaptsinos, D., Jalel, N.A. and Leigh, J.R. Estimation of state variables of a fermentation process via Kalman filter and neural network. In: *Application of Neural Networks to Modelling and Control,* (eds G.F. Page, J.B. Gomm and D. Williams), Chapman and Hall, London, 53-73, 1993.

Vaidyanathan, R. and Venkatasurbramanian, V. Representing and diagnosing dynamic process data using neural networks, *Eng. App. of Artificial Intelligence,* 5(1), 11-21, 1992.

Wang, J.   Multiple-objective optimisation of machining operations based on neural networks, *Int. J. Adv. Manuf. Technology,* 8(4), 235-243, 1993/

Wang, X.T., Du, F. and Yang, Q.X. A novel model of intelligent design system and its implementation techniques. In: *Applications of Artificial Intelligence in Engineering VIII,* Vol.1, (eds G. Rzevski, J. Pastor and R.A. Adey), Computational Mechanics, Southampton, 97-106, 1993.

Widrow, B. and Hoff, M.E.   Adaptive switching circuits, *Proc. 1960 IRE WESCON Convention Record,* Part 4, IRE, New York, 96-104, 1960.

Wu, H.J. Expert systems for tool wear detection via neural network approach, *Proc. World Congress on Expert Systems,* Vol.3, (ed J Liebowitz), Pergamon Press, Oxford, 2180-2185, 1991.

Yao, Y.L. and Fang, X.D. Assessment of chip formation patterns with tool wear progression in machining via neural networks, *Int. J. Machine Tools and Manufacture,* 33(1), 89-102, 1993.