Erciyes Üniversitesi Fen Bilimleri Enstitüsü Dergisi, **18** (1-2), 1-11, 2002

ARTIFICIAL NEURAL NETWORK APPLICATIONS TO CONTROL

Şahin YILDIRIM
Robotic Research Lab., Mechanical Engineering Dept., Engineering Faculty, Erciyes University, Kayseri-Turkey

Abstract:The use of Artificial Neural Networks (ANNs) for the control of robots has recently become an active research topic in control engineering. This paper is presented a review of current research in control of robotic systems based on ANN techniques. The main concepts of ANNs are presented and the mathematical background of supervised learning in multilayer networks is outlined in first section. A brief summary on adaptive control of dynamic systems is given second section. The application of ANNs to control systems is discussed in the last section.

Keywords: Neural network, robotic, control systems, adaptive control

KONTROLDA YAPAY SİNİR AĞLARI UYGULAMLARI

Özet Yapay Sinir Ağlarının (YSA), robotların kontrolu için kullanımı son zamanlarda kontrol mühendisliğinde etkili araştırma konusu olmuştur. Bu makalede, YSA esaslı robot sistemlerin kontrolunda yaygın araştırma incelenmesi sunulmuştur. İlk bölümde, YSA'nın temel kavramı ve çok katmanlı danışmalı öğrenmeli sinir ağının matematik ifadesi verilmiştir. Dinamik sistemlerin adaptif kontrolu kısa özet halinde Bölüm 2'de değinilmiştir. En son bölümde YSA'nın kontrol sistemlerine uygulanması detaylıca sunulmuştur.

Anahtar kelimeler: Sinir ağları, robot, kontrol sistemleri, adaptif kontrol.

1. Artificial Neural Networks

ANNs are made up of simple, highly interconnected processing units called neurons each of which performs two functions: aggregation of its inputs from other neurons or the external environment and generation of an output from the aggregated inputs. The output from a neuron is fed to other neurons to which it is connected via weighted links. Through this simple structure, ANNs have been shown to be able to approximate most continuous functions to any degree of accuracy, by the choice of an appropriate number of neurons and output activation.

1.1 Artificial Feedforward Neural Networks

Artificial Feedforward Neural Networks (AFNNs) are made up of one or more hidden layers between the input and output layers, as illustrated in Figure 1. The functionality of the network is determined by specifying the strengths of the connection paths called weights and the neuron activation function. The input layer distributes inputs to the first hidden layer. The inputs then propagate forward through the network and each neuron computes its output according to

$$xi^m = g\left(\sum_{j=1}^{n_{m-1}} W_{ij}^m \, xj^{m-1} + bim \right) \quad (1)$$

where xi^m is the output of the *i*th neuron in the *m*th layer, Wij^m is the weight of the connection between the *j*th

neuron of the *(m-1)*th layer and the *i*th neuron of the *m*th layer and bi^m is the bias of the *i*th neuron in the *m*th layer. bi^m can be regarded as the weight of the connection between a fixed input of unit value and neuron *i* in layer *m*. The function g(.) is called the neuron activation function.

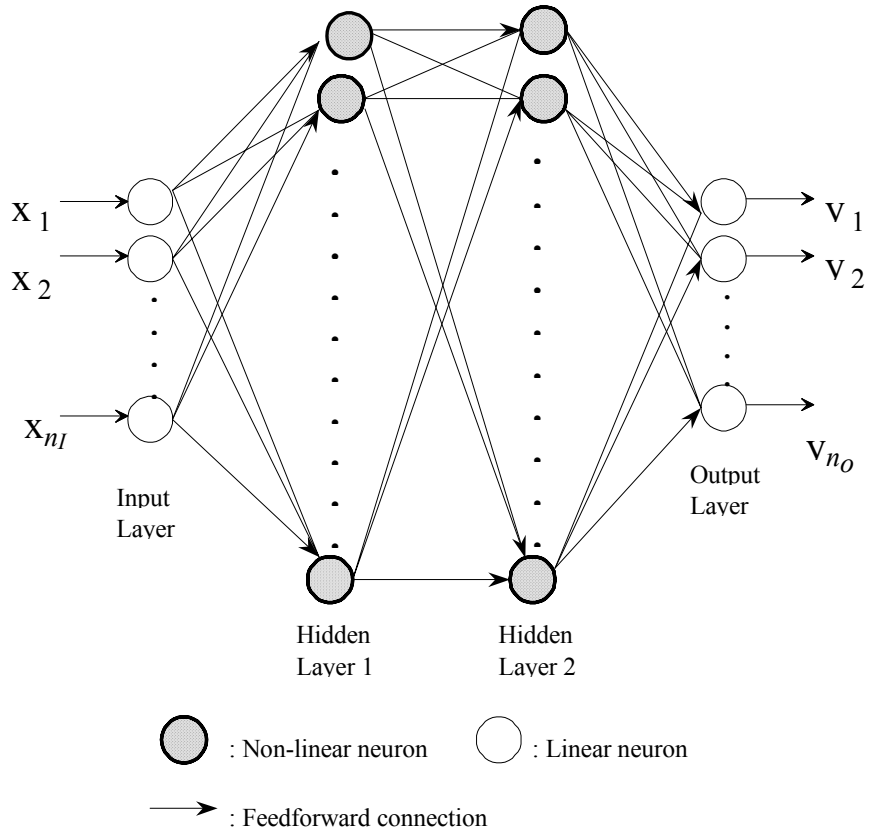

Figure 1. Feedforward network

The argument

$$z_i^m = \sum_{j=1}^{n_{m-1}} W_{ij}^m x_j^{m-1} + b_i^m \tag{1}$$

is the activation for the *i*th neuron in the *m*th layer. The function g(.) is assumed to be differentiable and to have a strictly positive first derivative.

For neurons in the hidden layers, the activation function is often chosen to be

$$g(z) = \frac{1}{1+e^{-z}} \tag{3}$$

Because the activation is non-linear, the neurons are said to be non-linear neurons. Since in system modelling applications the dynamic range of the output data may be greater than 1, the activation function of the output nodes is chosen to be linear and the output nodes are said to be linear neurons. Thus, the *i*th output node performs a

weighted sum of its inputs as follows:

$$y_i = \sum_{j=1}^{n_{m-1}} W_{ij}^{O} x_j^{m-1} \tag{4}$$

where *m* represents the output layer and W_{ij}^{O} is the weight of the connection between the *j*th neuron of the last hidden layer and the *i*th neuron of the output layer.

Only networks with one hidden layer are considered in the present study, because the results of Cybenko [1989] and Funahashi [1989] show that this is sufficient to approximate all continuous functions. Let $\Theta = [\theta 1 \ldots\ldots\ldots \theta n]^T$ represent all the unknown weights and the biases of the network, where *n* denotes the dimension of the parameter vector Θ and is defined as $n = n_H (nI+1) + nH\ nO$, where *n*I, *n*H, and *n*O refer to the number of neurons in the input layer, in the hidden layer and in the output layer respectively. The *i*th output of a network with a single layer of hidden units can then be defined by:

$$yi = \sum_{j=1}^{n_H} W_{ij}^{O} x_j^{H} = \sum_{j=1}^{n_H} W_{ij}^{O}\ g\Big(\sum_{k=1}^{n_I} W_{jk}^{I}\ xk^{I} + bj^{H} \Big) \quad 1 \le i \le nO \tag{5}$$

where $\mathbf{x}^{I} = [x_1^{I} \ldots\ldots\ldots xn_I^{I}]^T$ is the input vector to the network, W_{jk}^{I} is the weight of the connection between the *k*th neuron of the input layer and the *j*th neuron of the hidden layer and b_j^{H} is the bias of the *j*th neuron in the hidden layer.

1.2 Artificial Recurrent Neural Networks

Artificial Recurrent Neural Networks (ARNNs) are different from feedforward ones in that their structure incorporates feedback. In general, the output of every neuron is fed back with varying gains (weights) to the inputs of all neurons (fully connected network). The structure is inherently dynamic, which gives the network powerful representation capabilities.

A well-known dynamic recurrent neural network is the Hopfield neural network, which was originally reported in the context of an associative method for pattern recognition [Hopfield, 1982]. The Hopfield neural network is a single-layer fully connected neural network. Most of its original applications required that the network performs as a stable system with multiple asymptotically stable equilibrium points [Hopfield, 1986].

The next class of recurrent neural networks, known as recurrent Backpropagation (BP) neural networks, was introduced by Pineda and other researchers [Pineda, 1987; Pineda, 1988; Pearlmutter, 1990; Sato, 1990]. Their work has shown that the BP algorithm can be extended to arbitrary neural networks as long as they converge to stable states. The learning algorithm for these networks is usually called recurrent BP or generalised BP. Figure 2 depicts the architecture of a recurrent BP neural network.

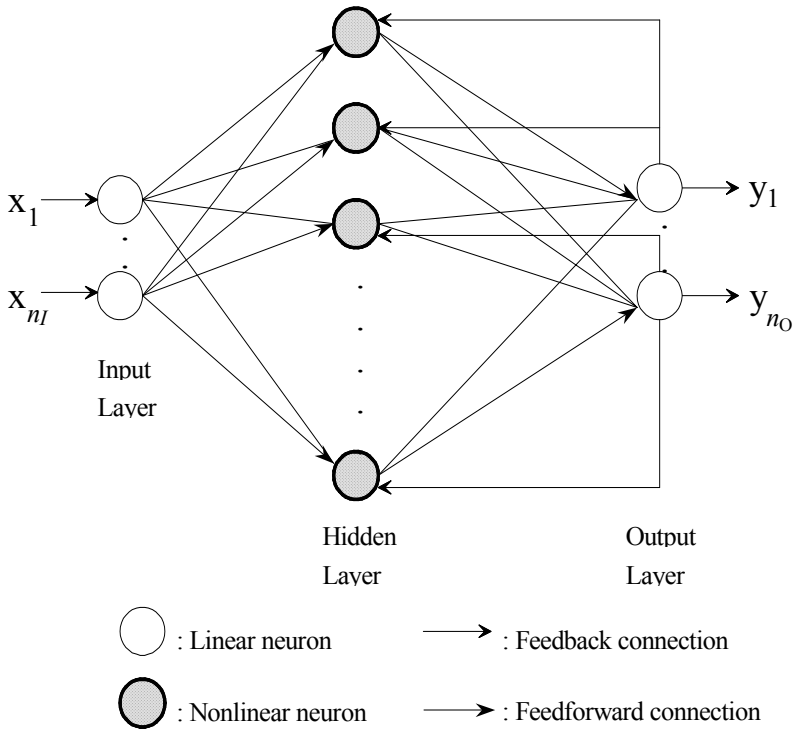

Figure 2. Configuration of a recurrent network

Another approach to learning a temporal sequence involves the so-called Tapped Delay Lines Neural Network (TDLNN), proposed by various researchers, including Lapedes and Farber [1987], Elman and Zipser [1988] and Narendra and Parthasarathy [1991]. However, there are drawbacks to the use of TDLNNs for sequence recognition such as the need for networks with many inputs and the consequent difficulty in training [Mozer, 1989].

Jordan [1986; 1989], Elman [1990] and Fuji and Ura [1991] have proposed "sequential" networks for temporal sequence recognition. In a network of this type, there are both feedforward and feedback connections. The recurrency due to the feedback connections gives the network a dynamic memory capacity. In most cases, the feedback connections have fixed weights and do not appreciably complicate the training and the BP algorithm may be used to modify the network weights at each time step. This type of recurrent network has been used in a number of applications [Jordan, 1989; Fuji and Ura, 1991].

2. Other Artificial Neural Networks

Various ANNs have been used to realise neural control schemes. Usually, the specific ANN that must be adopted will depend on the dynamic characteristics of the system. If the system is time dependent, the ANN must possess the capability to model temporal input-output relationships. Recurrent networks have inherent feedback and hence possess a suitable structure for temporal data. On the other hand, multilayer networks provide static non-linear mappings and hence have been used mostly for modelling time-invariant non-linear systems. The most popularly used paradigm is the multilayer feedforward network with the BP training algorithm. Although its original structure is for static representation, it can be adapted for dynamic modelling by introducing delay elements and feedback from the output layer to the input layer.

Another important ANN model is the Cerebellar Model Arithmetic Computer (CMAC). The CMAC network was proposed by Albus [1975a; 1975b] to model the functional methods that a physiological cerebellum uses to co-ordinate motor neurons in animals. CMAC is an associative memory network and can be used to model high-dimensional non-linear systems. CMAC is not limited to control problems and can be regarded as a model-free function estimator, like other types of ANNs. The stability and convergence properties of CMAC are also well established [An et al., 1994] and the growth of interest in CMAC, especially in robotic control applications, arises primarily from its desirable characteristics such as simple structure, Least Mean Square training rule, fast training speed and localised generalisation of knowledge.

2.1 Learning Algorithm

2.1.1 Backpropagation Algorithm

The Backpropagation (BP) algorithm is described in detail in Rumelhart and McClelland [1986]. Essentially, the algorithm includes propagating the differences between the desired and actual values of the outputs of the NN from the output layer through the hidden layer(s) to the input layer to compute the changes to the weights of the connections between the different layers. This is done so as to reduce those differences and make the NN produce the desired outputs.

2.1.2 Problems with the BP Algorithm

Despite many successful applications of the BP algorithm, there can be problems with networks adopting it [Wasserman, 1989]. The most troublesome problem is the long and uncertain training process, which sometimes even fails to converge. Convergence failures generally arise from network paralysis, local minima and temporal instability [Wasserman, 1989].

1. *Network Paralysis*: As the network trains, the values of the weights may become very large. This can force all or most of the neurons to operate at high output levels, where the derivative of the activation function (which is usually a sigmoid) is very small. Since the error propagated backwards for training is proportional to this derivative, the training process can be brought to a standstill. This is termed network paralysis. There is little theoretical understanding of this problem although various heuristics may be adopted to avoid it or to recover from its effects [Wasserman, 1989].

2. *Local Minima*: Local minima can arise if the error surface is in a convoluted and high-dimensional space. In such a situation, the network may become trapped in the neighbourhood of a local minimum, because BP is driven by the gradient of the surface which is flat at a minimum point.

3. *Temporal Instability*: This refers to the disruption of previously learnt information by new training data. The standard BP algorithm requires that all vectors in the training set be presented to the network before weight adjustment. Thus if new data were to be added, the training procedure would need to be repeated with the complete data set. The training process may oscillate if the network operates in a changing environment where it might never have the same input twice.

3. Adaptive Control Systems

The last decades have seen some extensive research and major breakthroughs in adaptive control [Craig et al., 1987]. The objective of adaptive control is to adjust the controller parameters in order to achieve a desired response of an unknown or time-variant system with or without an estimated model and/or reference model. If properly implemented, it can reduce the variability of parts being produced and increase productivity giving substantial economic benefits.

3.1 Traditional Adaptive Control Methods

Adaptive control has been a major topic of technical inquiry in the control community since the mid 1950s [Landau, 1979; Astrom, 1989]. Research in adaptive control was motivated by a desire to perform real-time control of partially or completely unknown systems. This attractive idea led to some early ingenious, but intuitive schemes. One of these initial approaches was simultaneously to estimate the unknown parameters of the system and generate control signals [Kalman, 1958]. However, these earlier schemes did not guarantee the system's stability. As system dynamics change, such a system can become unstable or lead to an unsatisfactory performance. Recently, parametric adaptive control strategies based on rigorous stability theorems have been developed to overcome the limitations of earlier schemes [Landau, 1979; Astrom, 1989].

Current adaptive control strategies can be divided into two categories: indirect and direct. The *indirect approach* is based on on-line system identification of the plant and simultaneous adjustment of control parameters. Self-tuning adaptive control strategies, which have become very popular, fall into this category. Self-tuning regulators seek to generate control laws for unknown systems, which would converge to the optimal strategies if the dynamics of the system are known.

In general, these adaptive controllers consist of three parts: a recursive parameter estimator, a design calculator and a feedback controller with adjustable parameters. Many different versions of self-tuning control are now available, mainly differing in controller design principles and on-line parameter estimation method. Some of the widely used design principles in self-tuning control entail pole-zero placement, minimum variance control and Linear Quadratic Gaussian Control (LQGC). One important requirement of the self-tuning control method is that on-line identification must yield an accurate estimate of the 'locally linearized' process model, when the system is non-linear, so that the control design strategy can generate accurate control parameters.

The Direct approach does not require explicit model identification. Instead, a desired process model is specified and the controller parameters are adjusted so as to minimise the error between the model and actual system outputs. Model Reference Adaptive Control (MRAC) falls into this category. Many different variations of this approach can be found in the literature [Landau, 1979].

3.2 Contributions of Artificial Neural Networks

Artificial Neural networks (ANNs) have demonstrated the capabilities of modelling a large class of non-linear systems and representing input-output relationships robustly. They can be trained to generate correct control signals and hence offer a great potential for adaptive control of non-linear systems. Also, the parallel processing nature of neural networks provides the capability of processing large amounts of information in real-time once a network has been trained.

The literature describes many types of neural adaptive controllers, corresponding to the conventional adaptive controllers outlined in current section. Narendra and Parthasarathy [1991, 1994] have presented several schemes for NN control. Different types of neural networks have been considered [Narendra and Parthasarathy, 1990]. The major underlying issue is how to train the network robustly and reliably when the system is rapidly time varying. The popular BP algorithm has been the primary choice for ANN training, in spite of the previously mentioned drawbacks.

3.3 Applications to Robotics

In the robotics community, there is currently a growing interest in using ANN technology [Ge et al., 1997; Er et al., 1997]. NNs offer several potential advantages over conventional control. For example, calculations are in principle carried out in parallel, yielding speed advantages, and programming can be done by training, rather than defining explicit instructions. Almost all ANN applications in robot control involve identifying the robot dynamics or inverse dynamics and incorporating this knowledge into the robot controller [Miller et al., 1990; Narendra and Parthasarathy, 1990; Pham and Oh, 1992; Pham and Yildirim, 1995c]. The approaches used differ in the methods of incorporating the NN into the controller and of training and adaptation. The basic idea is to employ a neural network to learn repeatedly characteristics of a robot and then use this knowledge to generate control inputs. The major advantage of the neural control approach is that it can produce a learning controller for a robot that can operate in an uncertain environment. A basic inverse model control scheme for a robot is shown in Figure 3.

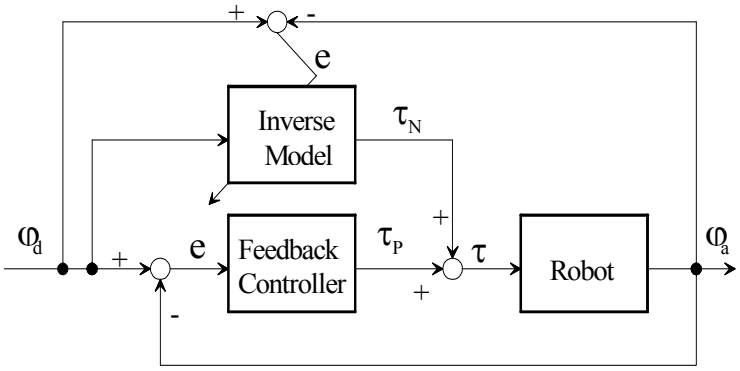

Figure 3. Inverse model control of a robot

3.4 Adaptive Artificial Neural-Control Architectures

Psaltis et al. [1988] proposed three different learning architectures for adaptive neural controllers: indirect learning, generalised learning and specialised learning.

The *indirect learning scheme* provides a means of generating a feedforward control law with the inverse model of the plant. A schematic of this learning architecture is shown in Figure 4. If a correct inverse model is established, the output of the overall

system can track the input commands closely. The training is carried out to minimise the error between the output of the artificial neural controller τ and the output of the inverse neural model of the plant τ_m. However, this scheme alone cannot lead to a satisfactory controller design, since the neural network model may not necessarily represent the actual system. This is because the network has the tendency of converging to a single value of τ when the input and output values of the plant are used for training.

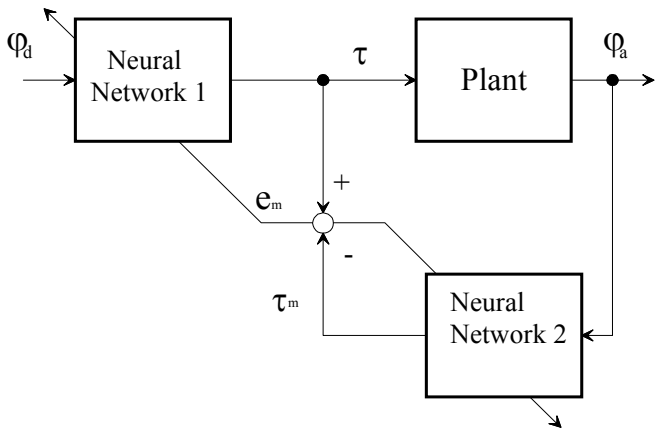

Figure 4. Indirect learning scheme

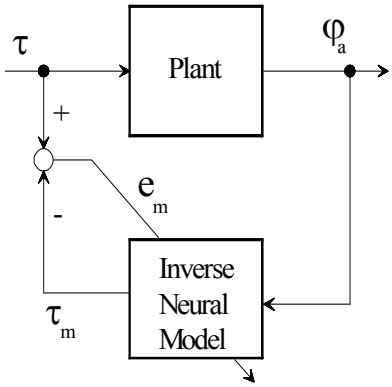

Figure 5. General learning scheme

The *generalised learning* method attempts to produce an inverse neural model for the plant by adopting a specified input space and generating corresponding output responses. This learning architecture is schematically shown in Figure 5. A major difficulty of this architecture is that one must choose an input set τ during training to generate outputs φa, which are sufficiently close to the desired input φd. This scheme, if used for adaptive control, can lead to an improperly trained network and may require persistent excitation, which can adversely affect the performance of a system [Psaltis et al., 1988].

The *specialised learning* method is an alternative method of training the network, as depicted Figure 6. Training involves using the desired responses φd as input to the network. The network then learns to find plant inputs τ that drive the system outputs φ_a to the desired φd, by using the error between desired and actual responses of the plant. This architecture addresses the main objections to the previous architectures; it can specially learn in the region of interest and it may be trained on-line by fine tuning itself, while actually performing useful work. The weakness of this architecture is that modifying the weights based on the total error must involve the plant, which normally is only approximately known.

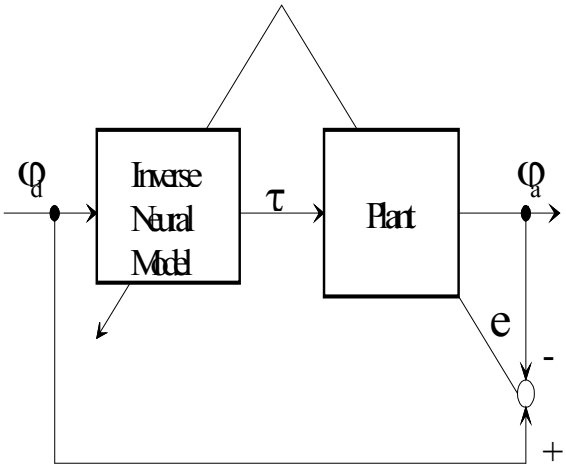

Figure 6. Specialised learning scheme

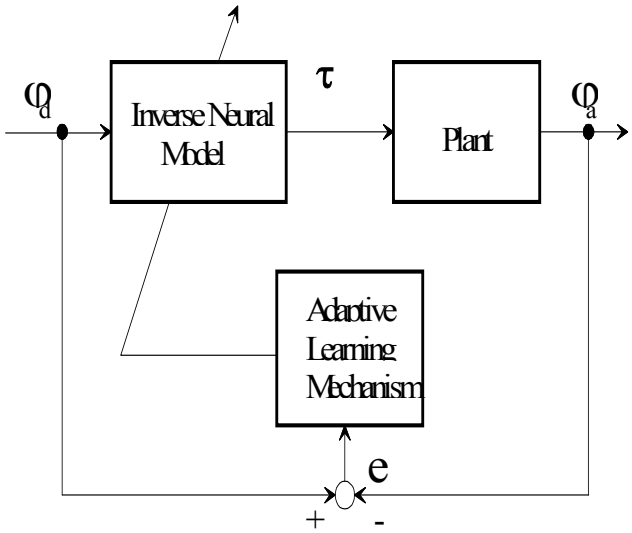

Figure 7. Open-loop feedforward neural adaptive control

The specialised learning scheme can be represented as an adaptive feedforward control scheme as shown in Figure 7. This type of controller, however, will have little ability to reject disturbances. If a feedback controller is used along with the feedforward controller, as shown in Figure 8, disturbance rejection will be improved. The feedback controller can be a conventional PID controller or a non-linear neural controller and the feedforward controller is a neural controller. As learning proceeds, the error signal will be reduced and the role of the feedforward controller increases.

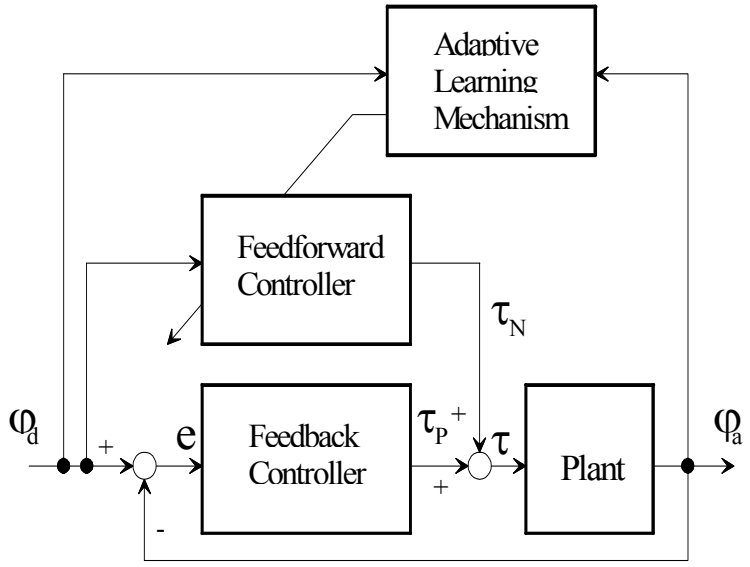

Figure 8. Feedforward adaptive control with neural network

A non-linear version of self-tuning control has been proposed that adopts a NN as a plant identifier [Chen and Pao, 1989], as shown in Figure 9.

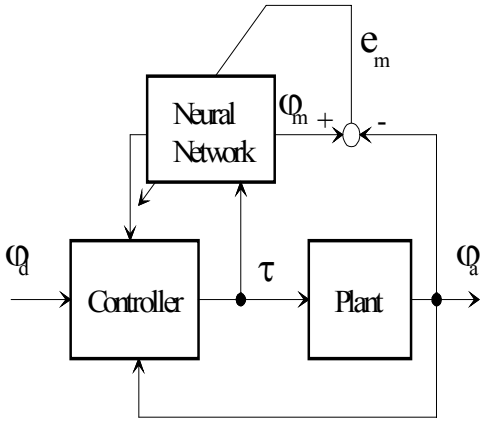

Figure 9. Self-tuning control with a neural network

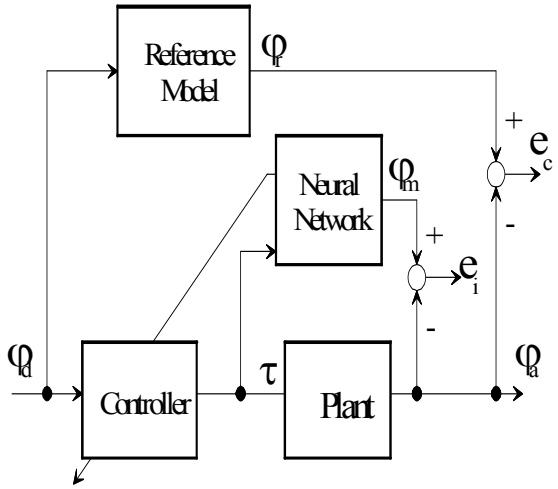

Figure 10. Indirect model reference adaptive control with neural networks

Similarly, a non-linear model reference adaptive control architecture employing a NN in place of the linear

controller has been described [Narendra and Parthasarathy, 1990]. In this case, the non-linear NN replaces the linear controllers. While both direct and indirect approaches have been used for the linear case, only the indirect method with a NN has been implemented so far. The non-linear plant dynamics is first identified by a NN and then the parameters of the controller are adjusted as shown in Figure 10.

4. Conclusion

This paper has discussed the basic concepts and characteristics of ANNs. Recent research in system identification and control using ANNs has been reviewed. General adaptive control schemes and various artificial neural control architectures have been described.

References

1. Albus, J.S., A New Approach to Manipulator Control: the Cerebellar Model Articulation Controller. Trans., ASME G, Journal of Dynamics Systems Measurement Control: 97, 220-227, 1975a.
2. Albus, J.S., Data Storage in the Cerebellar Model Articulation Controller. Trans., ASME G, Journal of Dynamics Systems Measurement Control, 97, 228-233, 1975b.
3. An, P.E., M. Brown, and C.J. Harris, Aspects of Instantaneous On-line Learning Rules. In Proc. of IEE International Conference on Control, 646-651, Coventry, UK, 1994.
4. Astrom, K.J., and B., Wittenmark. 1989. Adaptive Control. Reading, Mass: Addison-Wesley.
5. Chen, V. C. and Y. Pao., Learning Control with Neural Networks. IEEE Int. Control on Robotics and Automation. 3, 1448-1453. 1989.
6. Craig, J.J., P. Hsu, and Sastry, S.S., Adaptive Control of Mechanical Manipulators. Int. Journal of Robotics Research, 6, 16-28, 1987.
7. Cybenko, G., Approximations by Superposition of a Sigmoidal Function. Mathematics of Control, Signals and Systems, 2, 303-314, 1989.
8. Elman, J. L., Finding Structure in Time. Cognitive Science, 14, 179-211, 1990.
9. Elman, J. L., and D. Zipser, Learning the Hidden Structure of Speech. Journal of Acoustical Society of America, 83, 1615-1626, 1988.
10. Er, M. J., and K.C. Liew, Control of Adept One SCARA Robot Using Neural Networks. IEEE Trans. on Industrial Electronics, 44, 762-768, 1997.
11. Fuji, T. and Ura, T., Neural-Network-Based Adaptive Control Systems for AUVs , Engineering Application of Artificial Intelligence, 4, 309-318, 1991.
12. Funahashi, K., On the Approximate Realisation of Continuos Mappings by Neural Networks, Neural Networks, 2, 1983-1992, 1989.
13. Ge, S. S., C.C. Hang., and L.C. Woon, Adaptive Neural Network Control of Robot Manipulators in Task Space. IEEE Trans. on Industrial Electronics, 44, 746-752, 1997.
14. Hopfield, J.J., Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons. Proc. of the National Academy of Sciences, 81, 3088-3092, 1984.
15. Hopfield, J.J., and D.W. Tank, Computing with Neural Circuit: A Model. Science, 233, 625-633, 1986.
16. Jordan, M.I., Attractive Dynamics and Parallelism in a Connectionist Sequential Machine. Proc. 8th Annual Conference of the Cognitive Science Society, Amherst, Massachusetts, 531-546, 1986.
17. Jordan, M.I., Generic Constraints on Underspecified Target Trajectories. Int. Joint Conference on Neural Networks (IJCNN 89), Washington D.C., 1, 217-225, 1989.

18. Kalman, R.E., Design of A Self-Optimising Control System. Trans. ASME, **8**, 2, 123-162, 1988.

19. Landau, I.D. 1979. Adaptive Control: A Model Reference Approach. Marcel Dekker, New York, NY.

20. Lapedes, A., and R. Farber, How Neural Nets Work. Proc. Neural Information Processing System Conference, Denver, Colorado, 442-456, 1987.

21. Miller, W.T., F.H. Glanz., and L. G. Kraft, CMAC: An Associative Neural Network Alternative to Backpropagation. Proc. IEEE, **78**, 10, 1561-1567, 1990.

22. Mozer, M.C.A., Focused Backpropagation Algorithm for Temporal Pattern Recognition. Complex Systems, 3, 349-381, 1989.

23. Narendra, K.S. and S. Mukhopadhyay, Adaptive Control of Nonlinear Multivariable Systems Using Neural networks. Neural Networks, **7**, 5, 737-752, 1994.

24. Narendra, K.S. and K. Parthasarathy. 1990. Identification and Control of Dynamical Systems Using Neural Networks. IEEE Trans. on Neural Networks, 1, 4-27.

25. Narendra, K.S. and K. Parthasarathy, Gradient Methods for The Optimisation of Dynamical Systems Containing Neural Networks. IEEE Trans. on Neural Networks*,* **2**, 2, 252-262, 1991.

26. Pearlmutter, B.A., Dynamics Recurrent Networks. Technical Report, CMU-CS-90-196. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1990.

27. Pham, D.T., and S.J. Oh., A Recurrent Backpropagation Neural Network for Dynamic System Identification. Journal of Systems Engineering*,* 2, 213-223, 1992.

28. Pham, D.T and Ş. Yıldırım, Control of the Trajectory of a Planar Robot Using Recurrent Hybrid Networks. Int. Journal of Machine Tools&Manufacture: Design, Research and Application, 39, 415-429, 1999.

29. Pineda, F.J., Generalisation of Backpropagation to Recurrent Networks. Physical Review Letters, **59**, 19, 2229-2232, 1987.

30. Pineda, F.J., Dynamics and Architecture for Neural Computation. Journal of Complexity, 4, 216-245, 1988.

31. Psaltis, D., A. Sideris., and A.Yamamura, A Multilayered Neural Network Controller. IEEE Control Systems Magazine, 17-21, 1989.

32. Rumelhart, D.E. and J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Foundations*,* Cambridge, MA: MIT Press, 1986.

33. Sato, M.A., Learning Algorithm to Teach Spatiotemporal Patterns to Recurrent Neural Networks. Biological Cybernetics, 62, 259-263, 1990.

34. Shin, Y.C., Adaptive Control in Manufacturing. In Dağlı, C.H.(editor) Artificial Neural Networks for Intelligent Manufacturing. Chapman & Hall, London, 399-411, 1994.

35. Wassermann, P.D., Neural Computing: Theory and Practice. Van Nostrand, New York, NY, 1989.

36. Yıldırım, Ş., Bipedal Robot Control Using Recurrent Hybrid Neural Network. International Journal of Neural network World, 1-2, 57-73, 1999.