Neural Control Systems

Bernard Widrow and Françoise Beaufays *

Department of Electrical Engineering
Stanford University
Stanford, CA 94305-4055

1 Introduction

In the past decade, the field of neural networks has grown from a mere handful of researchers to thousands spread throughout academics and industry worldwide. Applications have been developed in domains as varied as character recognition, pattern classification, dynamic system control, and stock market prediction. In this paper, we concentrate on the application of feedforward multilayer neural networks to nonlinear control. After a brief introduction on adaptive linear combiners and multilayer neural networks, we present one of the most famous algorithms used in neural control: the backpropagation-through-time algorithm. A series of applications developed at Stanford University are discussed to illustrate the applicability of the algorithm. We then briefly summarize some recent theoretical work related to the subject matter, and we conclude by listing a couple of examples of neural controllers actually used in the industry.

2 Adaptive Linear Combiner and LMS Algorithm

The basic building block of a layered neural network is the *adaptive linear combiner* shown in Fig.1. This element receives an input vector or *input pattern*, and outputs a weighted combination of its components. During the training process, the adaptive linear combiner is presented with input patterns and corresponding desired output responses. At each presentation, an error signal is defined as the difference between the actual and the desired outputs, and is used by a *training algorithm* to adjust the weights. The simplest and most popular training algorithm for the linear combiner is the LMS (least mean square) algorithm [1, 2], also called the Widrow-Hoff delta rule [3]. It proceeds by iteratively modifying the weights in such a way as to minimize the sum of the squared errors over the training set. In other words, it defines an error surface over the weight space, and searches it for its minimum using a stochastic steepest descent method. The resulting algorithm consists simply of adjusting the weights at each pattern presentation by an amount proportional to the product of their input and the error signal.

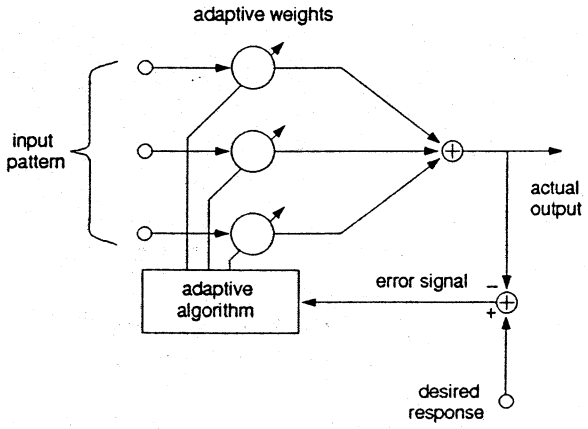


Figure 1: *Adaptive linear combiner.*

*This work was sponsored by NSF under grant NSF IRI 91-12531, and EPRI under contract RP8010-13.

The development of the LMS algorithm and its application to adaptive signal processing led to major commercial applications such as adaptive equalization in high-speed modems [4, 5] and echo cancellation for long-distance telephone and satellite circuits [6]. In the area of pattern classification, linear adaptive combiners were cascaded with hard-limiting thresholds to produce linear binary classifiers, and combinations of such elements were built to classify non-linearly separable patterns [7, 8]. Another family of adaptive structures based on the same principle was obtained by forming layered arrangements of linear combiners, each of them cascaded with an S-shape or *sigmoidal* nonlinear function. The resulting structure is the now famous feedforward multilayer neural network.

3 Feedforward Multilayer Neural Networks

A feedforward multilayer neural network is represented in Fig.2(a). It consists of successive layers of adaptive linear combiners or *neurons*, each of which is cascaded with a nonlinearity called a *sigmoid*. At each iteration of the training process, the neural network is presented with an input pattern and a corresponding desired response. Following the principle of LMS, an error surface defined over the weight space is iteratively minimized using a stochastic steepest descent method. Due to the layered structure of feedforward networks, steepest descent can be implemented in a very efficient way, the resulting algorithm is called *backpropagation* [9, 3].

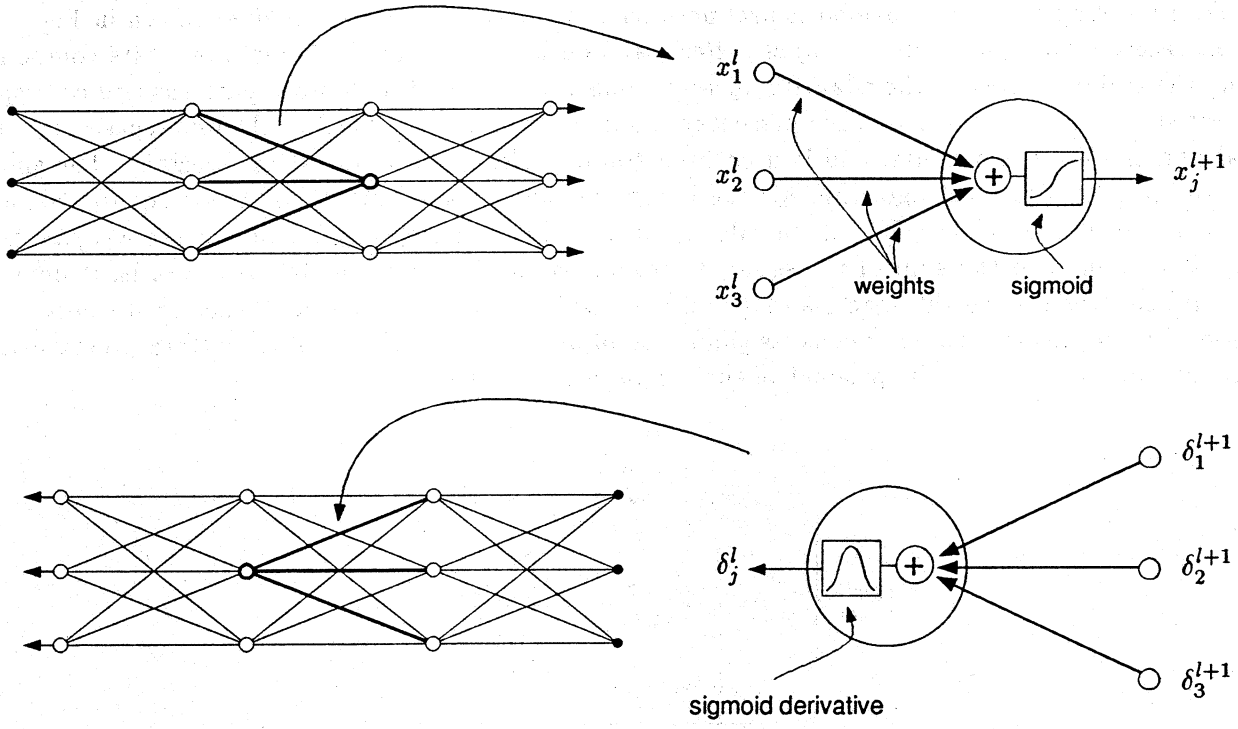


Figure 2: *Feedforward multilayer neural network and details of a specific neuron. (a) Forward propagation of the input vector* $\mathbf{x}$ *(b) Backward propagation of the error vector* δ.

Without proof, we will summarize the algorithm (a detailed derivation can be found in [3]). At each pattern presentation, the input vector is propagated forward through the network (see Fig.2(a)). The error vector is then propagated backward through a network identical to the original one but where the sigmoids have been replaced by their derivatives (see Fig.2(b)). The weights of each neuron are then updated by an amount proportional to the product of the inputs to that neuron and the error backpropagated to the

output of the neuron. Note the similarity between this algorithm and the single-neuron LMS algorithm.

An alternative to backpropagation is the MRIII algorithm. Intuitively, backpropagating the error vector through a given layer of the network is equivalent to multiplying the errors by the derivatives of the outputs of the layer with respect to its inputs. MRIII estimates these derivatives by independently perturbing each neuron by a small amount and comparing the output of the perturbed network with the unperturbed one. The equivalence between the weight updates performed by MRIII and backpropagation is demonstrated in [10]. Although MRIII presents some hardware implementation advantages over backpropagation, it usually requires more computations and is therefore less often used in practice.

4 Neural Control Systems and Backpropagation-Through-Time

One of the most promising new areas of application for layered neural networks is the control of non-linear dynamic systems. Among the different algorithms dealing with the adaptation of neural networks embedded in dynamic structures, the most widely used is *backpropagation-through-time* [9, 3, 11]. In an early application to neural control, the algorithm was used by Nguyen and Widrow [12, 13] to teach a computer-simulated truck to backup to a loading dock. The control structure used to solve this problem is shown in Fig.3. The truck kinematics are modeled by a set of discrete time equations that constitute the plant model. At each time step, the neural controller takes as input the state of the truck (i.e. its position and orientation) and computes a steering angle. The truck is then backed up a fixed distance, with the steering angle defined by the neural controller. The new state of the truck is computed by the plant model, and the operation is repeated until the truck hits near the dock. An error vector is then defined as the difference between the state of the truck and its desired state (centered and perpendicular to the dock).

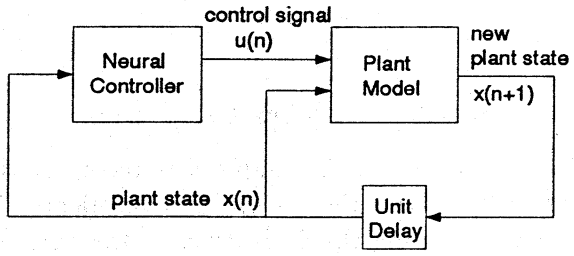


Figure 3: *Neural controller and plant model: block diagram.*

Note that no desired network outputs are made available to the learning algorithm. Only the desired final state of the truck is known. The approach used by Nguyen and Widrow to solve this problem is two-staged. First, a neural network is trained to emulate the kinematics of the truck, then the neural controller is adapted. Unravelling in time the feedback loop of Fig.3 and replacing the truck model by its neural emulator, we obtain a giant layered neural network (Fig.4) whose desired output is known. Backpropagation can be applied to the unravelled structure (hence the name backprop-through-time), and the weights of the controller can be adjusted accordingly.

After training, the truck was capable of backing up from almost any initial position, including initial positions never seen before. A sample trajectory is shown in Fig.5(a).

The same algorithm was then applied to teach a neural network to balance a pendulum mounted in equilibrium on top of a cart [14]. The neural network was constrained to output full-magnitude forces that were applied to the cart (bang-bang control). Again, a first neural network was trained to emulate the cart-broom discrete state-space equations, and a second network was trained to control the system. After adaptation, the neural controller was capable of bringing an initially-off-centered cart back to the center of the track while keeping the broom vertical.

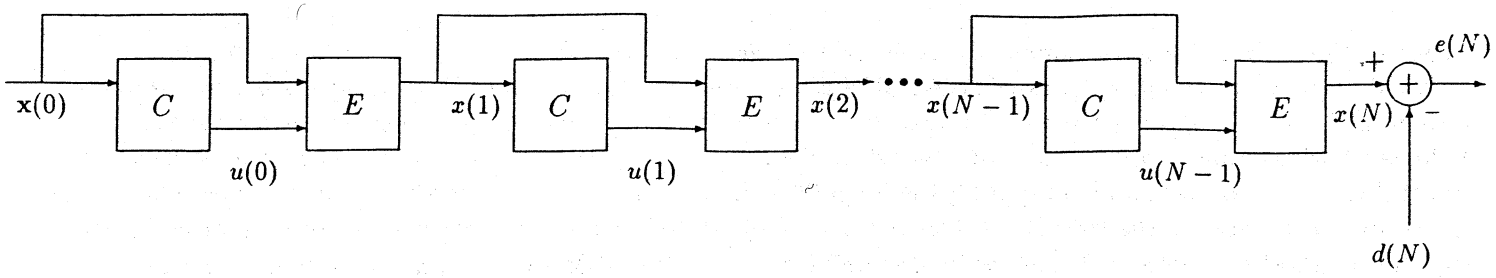
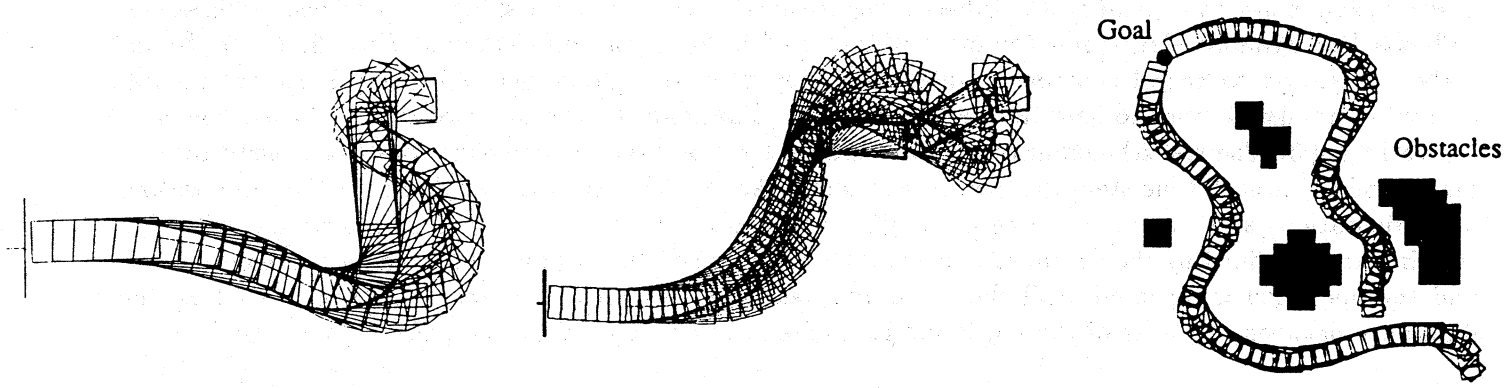


Figure 4: *Neural controller and plant model: block diagram unravelled in time. The C and E blocks represent the neural controller and emulator; $x(n), u(n)$ are the plant state and control signal at time n respectively; $d(N), e(N)$ are the desired response and the error signal at time N respectively.*



(a) Single-trailer truck

(b) Double-trailer truck

(c) Single-trailer truck around obstacles.

Figure 5: *Sample truck trajectories.*

An important theoretical step done by Widrow's group was to recognize that instead of backpropagating errors through a neural emulator of the plant, the errors can equivalently be multiplied by the Jacobian matrix of the plant, i.e. the matrix containing the derivatives of the outputs of the plant with respect to its inputs. Derivations of this result can be found in [15, 16]. Backpropagation directly through the plant model can be done whenever such a model is available, and results in a significant saving in development time. Based on this Jacobian method, a double-trailer truck was taught to backup to a loading dock (see sample trajectory in Fig.5(b)), and a double-broom-cart system was balanced.

As a further extension to the family of truck problems, a single-trailer truck was then taught to backup around obstacles [17, 18]. In solving this problem, a potential field is iteratively built around the obstacles (high values correspond to locations close to obstacles, low values to locations close to the dock). The truck then follows a steepest descent path in this potential field to reach the dock while avoiding the obstacles (see sample trajectory in Fig.5(c)).

In the area of power systems, backpropagation-through-time was used to solve an excitation control problem. In this application, a synchronous generator is connected to a load that can vary over time. As a result of load variations, the terminal voltage of the generator fluctuates. Proper action must be taken on the excitation or *field* current in order to bring the terminal voltage back to its nominal value after each load variation. Three neural networks were adapted to control the excitation current of the computer-simulated generator shown in Fig.6: one acting on the DC voltage of the excitor, one setting the position of a rheostat inserted in the excitor circuit, and one controlling both the excitor voltage and the rheostat. While the first control scheme could have been efficiently performed by a linear controller, the two neural controllers acting on the rheostat were solving nonlinear differential equations that would have

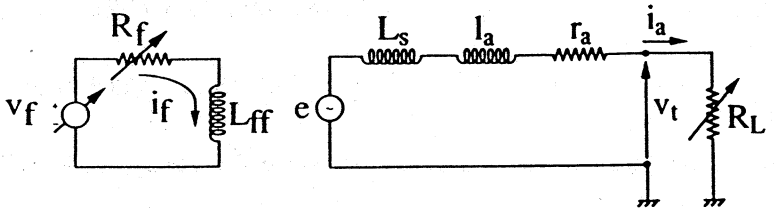


Figure 6: *Single generator driving a time-varying load: simplified model.* v_f, R_f, i_f, L_{ff} *are the DC excitation or* field *voltage, the excitation rheostat, the excitation current, and the rotor self-inductance respectively;* $e, L_s, l_a, r_a, i_a, v_t, R_L$ *are the stator induced emf, the armature reaction inductance, the armature leakage reactance, the armature resistance, the armature current, the terminal voltage, and the resistive time-varying load respectively.*

been difficult to solve by conventional methods. Among the three control schemes, simultaneous control of the DC voltage and rheostat has shown to allow the fastest terminal voltage recovery. Simulation results can be found in [19].

5 Recent Theoretical Work in Neural Control Systems

In terms of control theory, the control structure of Fig.3 can be seen as an extension of dynamic state feedback, where the feedback gains are replaced by a nonlinear neural network. Since the neural controller-plant system is nonlinear, notions such as deadbeat control can not be maintained. The number of time steps needed by the controlled system to reach its desired state is a parameter whose setting is left to the intuition of the control designer instead of being optimized by the neural controller. To overcome this problem, theory has recently been developed in our lab that shows how backpropagation-through-time can be rephrased as an extension of classical optimal control and how neural networks can be used to solve so-called "open final time problems" such as minimum-time control [20].

6 Neural Controllers in Industrial Applications

One could argue that all the examples presented so far are "toy-problems" and computer-simulated applications. We conclude this paper by listing a couple of examples of neural controllers actually implemented in industrial environments.

A first example, taken from the power systems industry, is a prototype system developed by BC Hydro in Vancouver. A neural network was taught to simultaneously control five synchronous condensors in such a way as to balance the imaginary power of an electric network without overstressing any particular machine. The set up successfully demonstrated the power of neural networks and illustrated the savings that they can bring in the power industry.

Another example is the Intelligent Arc Furnace Controller^{TM}, a neural controller developed by Neural Applications Corporation to position the electrodes of a steel melting furnace [21]. Replacing the rule-based system previously used, the neural controller was implemented at North Star Steel Co. It increased the furnace productivity, reduced the electrode consumption, and improved the electric power factor of the plant, resulting in major cost reductions.

Finaly, a more surprising application, the details of which are proprietary, is the use of backpropagation-type neural networks by M & M Mars to improve the productivity and reliability of candy manufacturing!

References

[1] B. Widrow and M. E. Hoff, Jr. Adaptive switching circuits. In *IRE WESCON Conv. Rec., pt. 4*, pages 96–104, 1960.

[2] B. Widrow and M. E. Hoff, Jr. Adaptive switching circuits. Technical Report 1553-1, Stanford Electron. Labs., Stanford, CA, June 30 1960.

[3] D. E. Rumelhart and J. L. McClelland, editors. *Parallel Distributed Processing.* The MIT Press, Cambridge, MA, 1986.

[4] R. W. Lucky. Automatic equalization for digital communication. *Bell Syst. Tech. J.*, 44:547–588, April 1965.

[5] R. W. Lucky, *et al. Principles of Data Communication.* McGraw-Hill, New York, 1968.

[6] M. M. Sondhi. An adaptive echo canceller. *Bell System Technical Journal*, 46:497–511, March 1967.

[7] B. Widrow. Adaline and madaline—1963, plenary speech. In *Proceedings of the IEEE First International Conference on Neural Networks*, volume I, pages 145–158, San Diego, CA, June 23 1987.

[8] B. Widrow and R. Winter. Neural nets for adaptive filtering and adaptive pattern recognition. *IEEE Computer*, pages 25–39, March 1988.

[9] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, Cambridge, MA, August 1974.

[10] D. Andes, B. Widrow, M. Lehr, and E. Wan. MRIII: A robust algorithm for training analog neural networks. In *International Joint Conference on Neural Networks*, volume I, pages 533–536, Washington, DC, January 1990.

[11] P. J. Werbos. Backpropagation through time: What it does and how to do it. *Proc. IEEE, special issue on neural networks*, 2:1550–1560, October 1990.

[12] D. Nguyen and B. Widrow. The truck backer-upper: An example of self-learning in neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume II, pages 357–363, Washington, DC, June 1989.

[13] D. Nguyen and B. Widrow. Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, April 1990.

[14] D. Nguyen. *Applications of Neural Networks in Adaptive Control.* PhD thesis, Information Systems Laboratory, Stanford University, Stanford, CA, 1991.

[15] S. W. Piche and B. Widrow. First-order gradient descent training of adaptive discrete-time dynamic networks. Technical Report RL-TR-91-62, Rome Laboratory, Air Force Systems Command, Griffiss Air Force Base, NY 13441-5700, May 1991.

[16] F. Beaufays, Y. Abdel-Magid, and B. Widrow. Application of neural networks to load-frequency control in power systems. *To be published in Neural Networks*, 1992.

[17] E. S. Plumer. Cascading a systolic array and a feedforward neural network for navigation and obstacle avoidance using potential fields. Technical Report CR-177575, National Aeronautics and Space Administration, Ames Research Center, Moffett Field, CA 94035-1000, February 1991.

[18] E. S. Plumer. Neural network structure for navigation using potential fields. In *Proceedings of the International Joint Conference on Neural Networks*, volume I, pages 327–332, Baltimore, MD, June 7-11 1992.

[19] F. Beaufays and B. Widrow. Application of neural networks to nonlinear control in power systems. *presented at the EPRI/NSF workshop on application of advanced mathematics to power systems*, September 1991.

[20] E. S. Plumer. Two neural network algorithms for designing optimal terminal controllers with open final-time. Technical Report CR-177599, National Aeronautics and Space Administration, Ames Research Center, Moffett Field, CA 94035-1000, July 1992.

[21] W. E. Staib and R. A. Staib. The Intelligent Arc Furnace^{TM} Controller: A neural network electrode position optimization system for the electric arc furnace. In *Proceedings of the International Joint Conference on Neural Networks*, Baltimore, June 1992.