

This explicit division of labor seems a reasonable strategy; while any problem-solving task can always, in principle, be solved with a sufficiently large set of (S,G)-R heuristics, it is also possible to learn the effects of the operators and the layout of the world as independent processes, which are only combined at the time of actual problem solving. By treating them as separate processes, more specialized, high-level techniques can be utilized for each task. In general, the ability to plan ahead and establish appropriate subgoals which are then fed to a simple "motor" system seems almost unavoidable. At some point, higher-level planning systems are necessary to augment the basic (S,G)-R/S-E system. The ability of the S-S system to set appropriate subgoals is one possible approach to that problem.

Thus, while a single, general mechanism can, in principle, learn correct behavior, the overall problem-solving task can also be approached as a number of more specialized tasks, each of which permits more specialized learning and representation mechanisms. By identifying specific aspects of the overall task and addressing them with specialized systems, performance can often be greatly improved. The complexity of the nervous system clearly reflects the variety of storage and processing tasks that can be productively addressed with specialized structures.

Road Map: Artificial Intelligence and Neural Networks

Related Reading: Planning, Connectionist; Reinforcement Learning in Motor Control

References

- Albus, J. S., 1981, *Brains, Behavior and Robotics*, Peterborough, NH: BYTE/McGraw-Hill.
 Barto, A. G., Sutton, R. S., and Anderson, C. W., 1983, Neuron-like

- adaptive elements that can solve difficult learning control problems, *IEEE Trans. Syst. Man Cybern.*, SMC-13:834-846.
 Bower, G. H., and Hilgard, E. R., 1981, *Theories of Learning*, Englewood Cliffs, NJ: Prentice Hall.
 Colwill, R. M., and Rescorla, R. A., 1986, Associative structures in instrumental learning, in *The Psychology of Learning and Motivation*, vol. 20 (G. H. Bower, Ed.), New York: Academic Press.
 Deutsch, J. A., 1960, *The Structural Basis of Behavior*, Chicago: University of Chicago Press.
 Gallistel, C. R., 1980, *The Organization of Action*, Hillsdale, NJ: Erlbaum.
 Hampson, S. E., 1990, *Connectionistic Problem Solving: Computational Aspects of Biological Learning*, Boston: Birkhäuser.
 Hampson, S. E., 1994, Problem-solving in artificial neural networks, *Progr. Neurobiol.*, 42:229-281.
 Jordan, M. I., and Jacobs, R. A., 1990, Learning to control an unstable system with forward modeling, in *Advances in Neural Information Processing Systems 2* (D. S. Touretzky, Ed.), San Mateo, CA: Morgan Kaufmann.
 Klopff, A. H., Morgan, J. S., and Weaver, S. E., 1993, A hierarchical network of control systems that learn: Modeling nervous system functions during classical and instrumental conditioning, *Adapt. Behav.*, 1:263-319.
 Mowrer, O. H., 1960, *Learning Theory and Behavior*, New York: Wiley.
 Rescorla, R. A., 1977, Pavlovian second-order conditioning: Some implications for instrumental behavior, in *Operant Pavlovian Interactions* (H. Davis and H. M. B. Hurwitz, Eds.), Hillsdale, NJ: Erlbaum.
 Samuel, A. L., 1963, Some studies in machine learning using the game of checkers, in *Computers and Thought* (E.A. Feigenbaum and J. Feldman, Eds.), New York: McGraw-Hill.
 Sutton, R. S., 1988, Learning to predict by the methods of temporal differences, *Machine Learn.*, 3:9-44.
 Werbos, P. J., 1990, Consistency of HDP applied to a simple reinforcement learning problem, *Neural Netw.*, 3:179-189.

Process Control

Lyle H. Ungar

Introduction

Neural networks are being used throughout the chemical-processing and related industries such as production of polymers, steel, and composites. Similar techniques are used in production of aircraft and many other industries. This article covers the use of neural networks for process control: not just control as it is typically defined (see ADAPTIVE CONTROL: NEURAL NETWORK APPLICATIONS and ADAPTIVE CONTROL: GENERAL METHODOLOGY) but a host of related applications such as virtual sensors and fault detection and diagnosis. It concentrates on examples in the control of chemical processes, but it should be of interest to people in a wide variety of industries with similar problems.

Neural networks have been used in a vast variety of different control structures and applications, serving as controllers and as process models or parts of process models (e.g., as virtual sensors). They have been used to recognize and forecast disturbances, to detect and diagnose faults, to combine data from partially redundant sensors, to perform statistical quality control, and to adaptively tune conventional controllers such as PIDs (proportional, integral, and derivative controllers). Many different structures of neural networks have been used, with feedforward networks, radial basis functions, and recurrent

networks being the most popular. Most of the examples cited here use standard feedforward networks; RADIAL BASIS FUNCTION NETWORKS (q.v.) are more popular in adaptive control, since they are linear in the coefficients of the basis functions (Sanner and Slotine, 1992).

The different uses of neural networks covered in this article can be characterized by a set of mappings as shown in Table 1. Neural networks may be used in direct control, where they provide a mapping from the current state (or, more commonly, current and recent sensor readings as in an ARMA model—see FORECASTING for definitions and details) and the desired next state of the plant to the control action to be taken. In indirect

Table 1. Input-Output Mappings for Different Uses of Neural Networks

Use	Input-Output Mapping
Direct control	Current state + Desired next state → Control action
Indirect control	Current state + Control action → Next state
Virtual sensors	Easy-to-measure properties → Hard-to-measure properties
Fault diagnosis	Sensor readings → Faults

control, the networks are used as a nonlinear model of the plant: given the current state of the plant and a proposed control action, they predict the next state of the plant. This model can then be used in a variety of controllers, such as model predictive control (MPC).

Neural networks can also be used as a piece of a larger model of the plant. Often one wishes to control variables such as concentration or viscosity, which are relatively hard to measure accurately. Other variables may be much easier to measure and are related to the variables of interest in a deterministic but highly nonlinear way. For example, a variety of spectral techniques such as infrared (IR) measurements can be used to predict concentrations of chemical species and hence quality properties such as viscosities. Neural networks can be trained on the basis of data collected in the laboratory to learn the mapping between the easier-to-measure and the harder-to-measure variables and then used as "virtual sensors." For example, it is often inconvenient and expensive to measure the exact chemical composition of the gases going up a stack, yet it is important to be able to monitor and control them to meet environmental regulations. Cheaper and more reliable sensors can be used to measure related variables, and then neural networks used to estimate the concentrations.

Finally, neural networks are being used to detect and diagnose faults and to compensate for sensor failures or drifts. These applications are closely related to control, serving either as preprocessors (filters) for controller inputs or as components in supervisory control schemes running above conventional process controllers.

This article covers three of the most important uses of neural networks in chemical process control: model predictive control, virtual sensors, and fault detection and diagnosis.

Model Predictive Control with Neural Networks

As mentioned previously, there are a number of ways that neural networks can be used in process controllers. In almost all cases, the strengths and weaknesses of the control scheme remain unchanged when a neural network is used in place of ARMA or mechanistic models. Many different neural network architectures can be used in neurocontrollers. The most popular are backpropagation networks and radial basis functions. In both of these cases, an ARMA style model is typically used in which the inputs to the neural network include lagged values of the relevant measured variables. Alternatively, when the process has variable time delays, it may be more advantageous to use internally RECURRENT NETWORKS (q.v.) and input just the current measured variable.

The *Internal Model Control* (IMC) framework (Garcia and Morari, 1982; Nahas, Henson, and Seborg, 1992) provides a good example of how neural networks can be incorporated into

controllers (Figure 1). In conventional IMC, a model of the plant, typically linear, is partially inverted to determine a control action. Neural networks can be used as the controller (for direct control), as the plant model (to be "inverted" for indirect control), or as both (Psychogios and Ungar, 1991).

For indirect control the model of the plant is learned by training a neural network so that its output \hat{y}_t approximates the function $y_{t+1} = f(y_t, u_t)$, where y_t is a set of plant measurements at time t , and u_t contains control actions. Lagged values of y and u may also be given as inputs to the network. Finding a control action requires partially inverting this model to find the control action u_t to minimize the difference $\|\hat{y}_{t+1} - y_{t+1}^{sp}\|$ between the plant state predicted at time $t + 1$, \hat{y}_{t+1} , and that specified by the set point y_{t+1}^{sp} . If the function f were linear, the minimum would be easy to find analytically. However, neural networks are nonlinear, and therefore Newton's method or similar nonlinear equation-solving methods must be used to find the optimal control action. Fortunately, the Jacobian (the derivative of f with respect to each of the inputs or, equivalently, the linearization of f) required for such methods is generally available, since it is required when using the backpropagation or conjugate gradient algorithms to determine the network weights.

As an alternative to the computationally burdensome partial inversion of the plant model, a neural network can be trained to directly approximate the desired control function:

$$u_t = g(y_t, y_{t+1}^{sp}) \quad (1)$$

This relationship provides a control law for a direct feedforward controller. In general, the model (or inverse model) learned will be imperfect, and feedback must be used to stabilize the system. IMC provides one framework for doing so, as shown in Figure 1. Consider the case where the model gives an estimate \hat{y}_{t+1} of the plant output which is higher than the actual output y_{t+1} . The difference $y_{t+1} - \hat{y}_{t+1}$, which is the estimation error, is subtracted from the set point y_{t+1}^{sp} so that the controller tries to achieve a lower target y . If the error in the model is roughly constant, this method will cancel the error due to model inaccuracy.

Under certain conditions, one can train the controller network directly from past plant data, but if the controller is used in an IMC framework, one must be sure it is an accurate inverse of the plant model, since although IMC compensates for approximate plant models, it does not compensate for inaccuracies in inverting the model. (If the plant model is not invertible, this approach will, of course, be hard to use.) One technique often used is to propagate errors back through the plant model to the controller (Jordan and Rumelhart, 1992; see SENSORIMOTOR LEARNING).

Although it is relatively easy to implement and widely used, IMC has several disadvantages. It is typically used as a one-

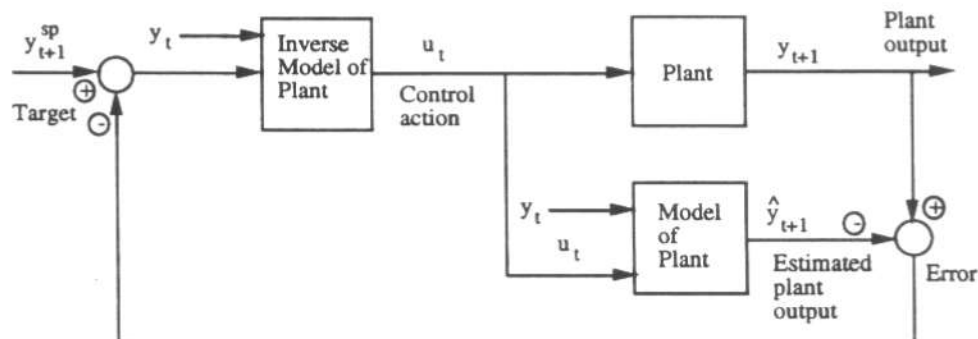


Figure 1. IMC control structure.

step-ahead predictor, and as such will not work on unstable plants or when there is a non-minimum phase response which requires looking more than one time step into the future to determine the correct answer. (For example, increasing the concentration of a reactant will often initially decrease the temperature and then later increase it; one-step-ahead controllers fare badly on such problems.) Also, IMC lends itself poorly to including constraints on the actuators (e.g., valves cannot open beyond some maximum) or on the controlled variables (e.g., temperature should not exceed some value). Thus it is more common to use neural networks in model predictive controllers.

The most widely used method of incorporating neural networks into controllers is within the framework of model predictive control (MPC), where the neural network is used as a process model (Psychogios and Ungar, 1991; Hunt et al., 1992). In this case, the method of MPC and all of its advantages and disadvantages remain unchanged; it simply becomes easier to obtain the model.

In a typical MPC architecture, one uses an optimizer to pick a sequence of control actions u to minimize the difference between the target y^{sp} and the actual value \hat{y} over the next N time steps. More formally, one wishes to pick u to find the minimum:

$$\min \sum_{i=1}^N a_i [y_i^{sp} - (\hat{y}_i - d)]^2 \quad (2)$$

where d is an estimate of persistent disturbances, used to provide a form of feedback to account for modeling errors, measurement errors, and process disturbances, and a_i is a weighting, typically giving more weight to errors in the near future.

Usually, one sets the control action constant over the last portion of the time horizon ($j = M$ to N).

$$u_j = u_{j+1} \quad j = M, \dots, N-1 \text{ for } N > M \quad (3)$$

As mentioned earlier, MPC allows one to set bounds on the output y , e.g., to keep a temperature in some range

$$y_{\min} \leq y_i \leq y_{\max} \quad i = 1, 2, \dots, N \quad (4)$$

MPC also allows one to set bounds on the control action u_i ,

$$u_{\min} \leq u_i \leq u_{\max} \quad i = 1, 2, \dots, N \quad (5)$$

and on the rate of change of the control action,

$$|u_i - u_{i+1}| \leq \Delta u_{\max} \quad i = 1, 2, \dots, M-1 \quad (6)$$

One then uses an optimization technique such as Sequential Quadratic Programming (SQP) to minimize Equation 2 subject to the constraints of Equations 3–6 and the neural network model of the plant,

$$\hat{y}_{i+1} = f(y_i, y_{i-1}, \dots, u_i) \quad (7)$$

Such optimization methods typically use gradient descent using the derivatives of the objective function (Equation 2) with respect to the control variables u_i , with Lagrange multipliers added to handle constraints. Note that if one is going to optimize over many steps, it is important to train the neural network to minimize the multistep prediction error rather than the one-step-ahead prediction error (see FORECASTING).

Neural networks are often offered as a panacea for the nonlinear models needed for MPC. After all, if one can take data from previous plant operation, fit a model, and hand it over to an optimizer, one should get a "perfect" controller. Unfortunately, this assumption is not true. Plant data are often inadequate to generate good models. In particular, data taken during closed-loop operation tend to provide a model of the controller rather than the plant. For example, if a plant has a

temperature controller on it, then when the plant temperature is above the set point, the controller will increase the flow of cooling water. Plant data will thus show that high cooling-water flow rates are correlated with high temperatures. A controller built using such a model will, of course, give disastrous performances. Similarly, but less spectacularly, if a given input has remained constant in the past, a neural network model (like any other regression-style model) will "show" that the input has no effect on the output. This result, too, will not provide a good basis for control.

Another problem with neural network models is that they are not necessarily accurate; insufficient data can produce models which are aphysical, even when an adequate network structure is built which includes the necessary delays of inputs. For example, they may not satisfy conservation of mass or energy, particularly when they are extrapolating or interpolating in regions where few data were available for training. Networks may give accurate predictions one time step in the future, but the predictions many time steps in the future used by the model predictive controller may still be inaccurate (see FORECASTING). More subtly, the models may be reasonably accurate in making predictions in the time domain but may give poor control because they are inaccurate in the frequency domain. Accurate models in the time domain (where neural networks are almost exclusively used) do not guarantee accurate models in the frequency domain, and frequency domain accuracy can be far more important for accurate control of dynamic systems. (Stability of controllers on a plant is determined by their dynamic characteristics in key frequency ranges, which may not be obvious when looking at standard plots of plant state over time.)

It bears repeating that although neural networks ease the task of building accurate multivariable nonlinear models, they do not eliminate the problems and difficulties involved in developing accurate and robust controllers. Although neural networks give good nonlinear models (and are much harder to analyze than linear models), their use is subject to all the usual limitations and dangers of conventional models. One must still determine appropriate control structures based on the plant and disturbance structure, including time delays and frequency response characteristics. IMC structures are not appropriate for controlling unstable systems, and accurate plant models are of no use if the disturbances dominate process behavior or if the plant is inherently hard to control because of poor design.

Hybrid Networks, Virtual Sensors, and Fault Diagnosis

Neural networks can also be used in conjunction with first principles models (Psychogios and Ungar, 1992) or as virtual sensors (Piovoso and Owens, 1991). Often one has a reasonable partial theoretical model of a plant but lacks certain parts of the model such as the dependence of viscosity or reaction kinetics on temperature and concentration. Neural networks can be used to learn that portion of the model. The overall hybrid model contains both the theoretical equations such as mass and energy balances and the neural networks, and can be used in an MPC scheme as described in the preceding section.

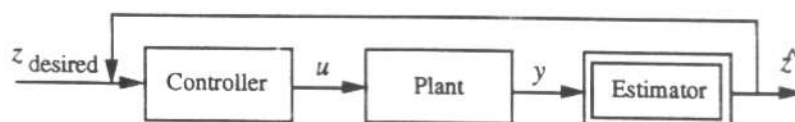
More formally, plants can often be described in the form

$$dx/dt = f(x, u, p) \quad (8)$$

$$p = g(x, u) \quad (9)$$

where the function f is known (e.g., kinematics or mass and energy conservation) but contains "parameters" p which depend on the plant state x and possibly the control action u in unknown ways. Observations of x and u can then be used to train a neural network to approximate the unknown function

Figure 2. Plant and controller using estimator as virtual sensor for z .



using an extension of the backpropagation algorithm even though the parameters p cannot be measured. The resulting plant model consisting of Equations 8 and 9 can be trained more accurately, using less data, and extrapolates better than the neural network plant models which do not incorporate the known form of f (Psychogios and Ungar, 1992).

A similar situation arises when one cannot easily measure some quality variables (e.g., viscosity, photodegradation, or concentrations of certain chemical species). For example, consider the control of the viscosity of a polymer product, where viscosity cannot be measured on-line. Temperatures y in the reactor can easily be measured, and the measurements can perhaps be supplemented with measurement of near-IR absorption. On-line measurement of viscosity is difficult. Often companies take samples, which are then sent to a laboratory for analysis. It can take from a half hour up to several hours to obtain an analysis from the laboratory. A neural network trained on product samples whose viscosity z has been measured in the lab can be trained to predict z as a function of variables y such as temperature and pressure which are measured on line:

$$z = f(y) \quad (10)$$

Such neural networks can be used as *virtual sensors* to give on-line estimations of viscosity for a controller.

Willis et al. (1992) give a nice example of the use of virtual sensors for inferential control of penicillin production. To optimally control penicillin production, one would like to have accurate on-line estimates of biomass concentration during fermentation. Such measurements are unfortunately not available. Willis et al. use a neural network to estimate biomass from carbon dioxide evolution rate, batch age, and the rates that the two primary substrates are fed to the reactor. These plant measurements (y in Figure 2) are then used to estimate the biomass z , which is the controlled variable.

Neural networks have also been used to minimize the effect of sensor noise and drift (data reconciliation), as elements of nonlinear statistical process controllers (SPC), and for fault diagnosis, sometimes as a prelude to reconfiguring control loops to cope with faults such as sensor or actuator failure. Neural networks can be trained which take a set of sensor readings as inputs and predict the same sensor readings as outputs (Kramer, 1992). When fewer hidden nodes are used than inputs, this system has the effect of nonlinearly projecting the sensor readings onto a smaller dimensional space and then reconstructing the sensor readings, thus reducing the noise in the signal. Such rectified sensor readings take advantage of the correlation structure of the sensor readings and, like other forms of filtered signals, can reduce the effect of sensor noise on control.

Neural networks have also been used extensively for fault diagnosis. In theory, it is relatively simple to train a network on a set of data where the inputs to the network are sensor readings just after a fault and the outputs of the network indicate what the fault was (Venkatasubramanian and Chan, 1989). A typical output-encoding scheme is to have one output for each fault, and have 1 represent the presence of a fault and 0 represent the absence of a fault. This design works well on simulated faults, but it can be more problematic in the real world. Apart from the usual questions of how much past (lagged) data to

include, in many plants there is not a sufficient quantity of data recorded from real faults, so one is forced to rely on less realistic data from simulators. Faults are often of different magnitudes (e.g., a leak may be of different sizes) and may occur when the plant is in different states (e.g., running at capacity or during start-up), and thus extensive fault data are required. Most data sets are for the case of single faults. It is only sometimes the case that a network trained on individual faults will be able to diagnosis combinations of simultaneous faults.

Discussion

The field of neural networks for process control is much too extensive to be fully summarized here: hundreds of papers are published each year in the area, and several commercial vendors offer control systems using neural networks. It is clear that neural networks are attractive models to use when processes are nonlinear and good first principles (mechanistic) models are not available, either for entire processes or for parts of the process. Most control schemes using neural networks are identical to those that do not use neural networks; the "neurocontrollers" just use neural networks as process models (e.g., in MPC), as virtual sensors, or as controllers. Neural networks can also be used to forecast disturbances (see FORECASTING) as part of a feedforward control system. Often, neural controllers are used as a feedforward component of a more complex control architecture in which a standard feedback controller rejects disturbances and gives adequate control while the neural network is being trained (see, e.g., Lee and Park, 1992, and citations there to Kawato and others). Many other uses of neural networks are possible, including their use as data preprocessors, as fault detectors (possibly coupled to a system which reconfigures the control loops when faults are discovered), and as automatic tuners for conventional controllers. Looking farther into the future, neural networks will be used to provide continuous optimization of plants, perhaps using REINFORCEMENT LEARNING (q.v.; see also Barto, 1990).

Road Map: Applications of Neural Networks

Background: 1.3. Dynamics and Adaptation in Neural Networks; Motor Control, Biological and Theoretical

Related Reading: Applications of Neural Networks; Model-Reference Adaptive Control; Reinforcement Learning in Motor Control

References

- Barto, A. G., 1990, Connectionist learning for control, in *Neural Networks for Control* (W. T. Miller, R. S. Sutton, and P. J. Werbos, Eds.), Cambridge, MA: MIT Press, pp. 5-58. ♦
- Bhat, N., and McAvoy, T. J., 1990, Use of neural nets for dynamic modeling and control of chemical processes, *Comput. Chem. Engrg.*, 14:573-583. ♦
- Garcia, C. E., and Morari, M., 1982, Internal model control. 1: A unifying review and some new results, *Ind. Eng. Chem. Process. Des. Dev.*, 21:308-323.
- Hunt, K. J., Sbarbaro, D., Zbikowski, R., and Gawthrop, P. J., 1992, Neural networks for control systems—A survey, *Automatica*, 28: 1083-1112. ♦
- Jordan, M. I., and Rumelhart, D. E., 1992, Forward models: Supervised learning with a distal teacher, *Cognit. Sci.*, 16:307-354.

- Kramer, M. A., 1992, Autoassociative neural networks, *Comput. Chem. Engrg.*, 16:313-328.
- Lee, M., and Park, S., 1992, A new scheme combining neural feed-forward control with model-predictive control, *Am. Inst. Chem. Eng. J.*, 38:193-200.
- Nahas, E. P., Henson, M. A., and Seborg, D. E., 1992, Nonlinear internal model control strategy for neural network models, *Comput. Chem. Engrg.*, 16:1039-1057.
- Narendra, K. S., and Parthasarathy, K., 1990, Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Netw.*, 1:4-27.
- Piovoso, M. J., and Owens, A. J., 1991, Sensor data analysis using neural networks, in *Chemical Process Control: CPC IV, Proceedings of the Fourth International Conference on Chemical Process Control*, New York: AICHE, pp. 101-118.
- Psichogios, D. C., and Ungar, L. H., 1991, Direct and indirect model based control using artificial neural networks, *Ind. Engrg. Chem. Res.*, 30:2564-2573.
- Psichogios, D. C., and Ungar, L. H., 1992, A hybrid neural network: First principles approach to process modeling, *Am. Inst. Chem. Eng. J.*, 38:1499-1511.
- Sanner, R. M., and Slotine, J.-J. E., 1992, Gaussian networks for direct adaptive control, *IEEE Trans. Neural Netw.*, 3:837-863.
- Venkatasubramanian, V., and Chan, K., 1989, A neural network methodology for process fault diagnosis, *Am. Inst. Chem. Eng. J.*, 35:1993.
- Willis, M. J., Montague, G. A., Di Massimo, C., Tham, M. T., and Morris, A. J., 1992, Artificial neural networks in process estimation and control, *Automatica*, 28:1181-1187.

Programmable Neurocomputing Systems

Nelson Morgan

Introduction

Neurocomputing is processing in which information is represented in the pattern of activity and connections between elements that perform simple operations on their inputs (e.g., a saturating nonlinearity of a weighted sum of the inputs). One of the attractions of neurocomputing is that it is often quite naturally parallelized. For this reason, much recent research in neurocomputing hardware has focused on efficient implementation of the most common neural algorithms, such as Hopfield networks, self-organizing feature maps, and backpropagation learning for multilayer perceptrons. Typically, matrix operations are at the core of these algorithms, and neural hardware is often customized for their efficient implementation. This can result in chips that are cost-effective solutions to specific applications, assuming that the design is also optimized for memory and I/O requirements.

Other research in neurocomputing is focused on developing new algorithms. For most of this work, specialized neurocomputing chips are not sufficient, and programmable computers are required. Workstations provide the necessary programming flexibility, but they are much slower than the best specialized chips. It is likely that specialization will always provide a speed advantage over the fully general approach.

Fortunately, computers can be more general than a single-function engine yet more specialized than a general-purpose computer. Architectures can be designed to be optimal for a class of applications and still have greater flexibility than a fixed-function implementation. Some common characteristics of these computers are:

- Single-task machine (no multi-user operating system)
- Physical memory only (no virtual memory)
- Arithmetic elements, interconnection, and memory paths designed to be efficient for a given class of algorithms
- Relatively short wordlengths (often fixed-point or integer) that are sufficient for the target tasks
- Some programmability

Neurocomputing hardware will nearly always include a fast dot-product engine that will be optimized for fixed-point or single precision floating-point operations. This is in contrast to both conventional supercomputers requiring double-precision

floating-point, and experimental (e.g., analog) neural chips using low-precision computational mechanisms. Otherwise, neurocomputers vary widely in the extent of specialization for neural computation. All can be programmed, but some with great difficulty, since the focus is on accelerating a limited form of computation. For instance, the matrix/vector operations that characterize much neural computation can often only be modified by writing microcode or nonstandard assembly language routines. In other systems, higher-level languages and commercial assemblers can be used for most modifications. The latter case needs additional resources for the control required for generality. However, a simple Reduced Instruction Set Computing (RISC) CPU can be fabricated on a compact piece of silicon, and can provide general-purpose programming capabilities. In some way, neurocomputers must permit users to program non-neural pieces of a full task.

This article provides a 1994 snapshot of programmable neurocomputers. Factors affecting utility and performance of these machines are discussed, with a look toward future machines. In fact, neurocomputer design is not fundamentally different from mainstream computer design; the ideas presented here should look familiar to the reader versed in conventional computer architecture (see Hennessy and Patterson, 1990, for a good text on this topic).

The focus in this article is on computing systems used for research and development of neural algorithms, as opposed to systems for embedded applications of production-ready networks, or research into network mappings onto the physics of electronics or optics. In this context, the word *programmable* refers to the ability of the neurocomputer to run programs, as opposed to the mutability of neural parameters such as weights. Finally, the applications focus is on network training for cases in which workstation technology is insufficient, for instance, with large amounts of training data and a large number of trained parameters.

System Requirements

Neurocomputation

While other operations are also necessary, fast computation of matrix products is the primary requirement for neurocomputing. Fundamental equations for search, learning, and recall