Pharmaceutical Research, Vol. 10, No. 2, 1993

Review

Introduction to Backpropagation Neural Network Computation

Randall J. Erb[1]

Neurocomputing is computer modeling based, in part, upon simulation of the structure and function of the brain. Neural networks excel in pattern recognition, that is, the ability to recognize a set of previously learned data. Although their use is rapidly growing in engineering, they are new to the pharmaceutical community. This article introduces neurocomputing using the backpropagation network (BPN).

KEY WORDS: neural network; neurocomputing backpropagation; modeling; pattern recognition; pharmacodynamics; pharmacokinetics; classification.

INTRODUCTION

Neurocomputing is concerned with parallel, distributed, adaptive information processing systems. These systems develop information processing capabilities in response to exposure to information (1). This computer modeling concept is based, in part, on how biologists believe the brain learns to recognize patterns. These computer models are known generically as neural network models. The models that compete with parametric statistical models excel at complex pattern recognition or classification.

Although there are many neural network models, the model that predominates in the area of pattern recognition or classification is the feedforward/backpropagation network (Fig. 1). This model is often referred to simply as a backpropagation network, BPN. The BPN was originally introduced by Werbos in 1974 (2) and owes much of its development to Rumelhart (3).

THE BPN MODEL

Neural networks are characterized by architecture, transfer function, and learning paradigm.

Architecture

The BPN architectural design consists of fully interconnected rows of processing units called nodes (Fig. 1). In neural network architectural structures, nodes are organized into groups called layers. Input layers receive input. Output layers produce output. Internal (or hidden) layers provide the interconnections between input and output. The net input into the jth layer node ($i[j]$) equals the sum of the outputs from the prior ith layer ($o[i]$). Each input from a prior node is multiplied by a weighting factor ($w[ji]$) associated with that particular interconnection;

$$\text{net input to a node} = i[j] = \Sigma_i \{w[ji]\ o[i]\}$$

The BPN is fitted by training the network with known input/output data sets, sometimes referred to as facts. The training paradigm finds a set of weight values that minimizes the error across the set of facts. During training, differences between actual outputs and model predicted outputs are propagated back through the architectural structure of the network. When the network has been optimized, as determined by convergence criteria, a new set of facts, called a test set, is fed forward through the network to evaluate the trained network. If the network is validated, it can then be used to predict outputs based on new input values.

BPNs can have multiple layers between the input layer and the output layer. In practice, however, many networks consist of only one hidden layer. This is because one layer is usually sufficient to provide adequate mapping even for continuous outputs (4–6). One hidden layer is enough for most pattern classification problems. In classification problems, the output node with the largest value is the most probable outcome for the given input pattern. An example of such a network is represented in Fig. 2. In this example, the pattern of EEG parameter inputs is used to determine the most probable drug classification. Many different EEG parameters can be used as inputs to predict the classification of the drug. Each output node represents one drug class. The output node that has the highest value is the drug class most likely to have produced the array of EEG inputs.

If outputs need to be continuous functions of the inputs, the model's ability to fit the data might be improved by using more than one hidden layer (7). An example of this input/output relationship is shown in Fig. 3. The network estimates the pharmacological effect profile (output vector) directly from the drug plasma profile (input vector). Each input node represents a plasma concentration at a specified

[1] Clinical Research Foundation—America, 11250 Corporate Avenue, Lenexa, Kansas 66219.

0724-8741/93/0200-0165$07.00/0

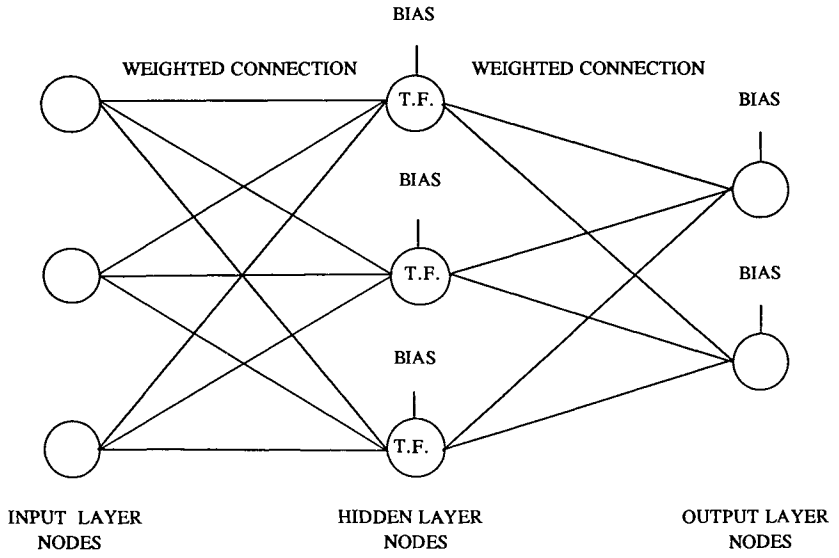

Fig. 1. Neural network schematic for a backpropagation neural network. Input into any node is determined by the weighted sum of the preceding layer of nodes, including a constant (bias) term. Nodal output is calculated using a sigmoid transfer function (T.F.).

time. Each output node represents a pharmacological effect at a specified time. Note that the sampling times for the input and output vectors need not be the same. Thus, time constants, such as those describing hysteresis, are built into the model with the training of the network.

The number of hidden nodes in a network can be critical to network performance. If a hidden layer has too few nodes, the network will lack the power it needs to classify patterns in the data. If a hidden layer has too many nodes, patterns will be memorized. Memorization handicaps the network's ability to generalize. The network has so many degrees of freedom that it responds with memorized facts rather than an estimated value based upon the general features of the facts (7). Memorization is analogous to fitting a set of N data points with an N-degree polynomial. A perfect fit results, but the fit's ability to interpolate is diminished.

In networks, complete memorization occurs when the number of hidden nodes equals the number of facts used to train the network. Kolmogorov's theorem predicts that

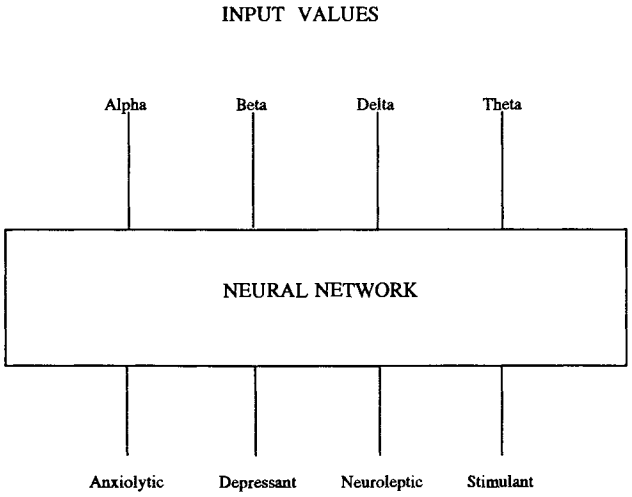

Fig. 2. Diagram of an example network designed to classify an input pattern. Input values are EEG values. Each output node has a value restricted between 0 and 1. The output node with the highest value is the most probable outcome.

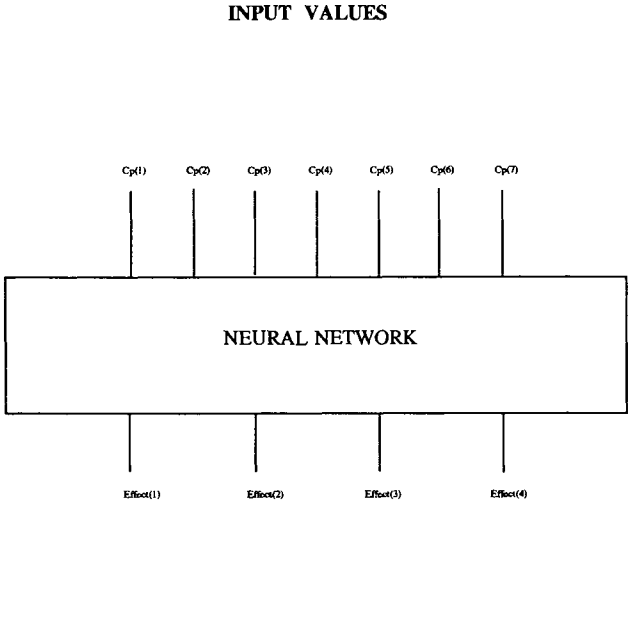

Fig. 3. Diagram of an example network designed to produce a continuous output relationship. In this example, each input node is the drug plasma level at a specified time. Each output node represents the effect of the drug at a specified time. The times of the output nodes do not have to correspond to the input nodes.

twice the number of input nodes plus one is enough hidden nodes to compute any arbitrary continuous function (8). So how does one determine the optimal number of hidden nodes in a network? One strategy is to start with Kolmogorov's number of hidden nodes, then reduce the number of nodes until a generalizing, working network is obtained (7). Linear regression can be used to evaluate the fit of the model. Unfortunately, "generalizability" has no comparable metric.

Transfer Function

The transfer function is a function that determines the output value of the node based on the total value of its inputs. Although several functions have been used as transfer functions, the most widely used is the sigmoid function shown in Fig. 4. The sigmoid function output from each jth hidden layer node ($o[j]$) is characterized by the relationship

$$\text{output}[j] = o[j] = 1/\{1 + \exp(-i[j])\}$$

where $i[j]$ is the sum of the inputs to the node. This function has two attributes that make it particularly well suited as a BPN transfer function. It is nonlinear in nature, and it is a function that limits output to values between 0 and 1. This sigmoidal normalization process provides for nonlinear outputs. It also prevents domination or overload effects in the network that could result from single large input values.

Another advantage of the sigmoid function is that it facilitates rapid network learning. This is because learning changes occur at the greatest rate midway in the 0–1 range, where the derivative (slope) is at its maximum (see Training Paradigm).

It is notable that during network training the sigmoid transfer function tends to seek an output value near 0 or 1. This represents the computer equivalent of a nonfire/fire status of a nerve cell. The phenomenon reinforces the computer model's relationship to the biological nervous system.

Some networks, such as the one shown in Fig. 1, have a bias term added to each node. These bias terms are a special form of the connection weights that are also optimized during the training process. This constant term for each node helps scale the average input into a usable range (7).

Training Paradigm

Training a network begins by randomly assigning weighting factors for each node interconnection and bias term. Next a fact is input through the network to produce an output value(s) based upon the randomly assigned weight values. New interconnect weights are then computed to reduce the total error until a minimum total error value is obtained across the set of training facts. A new weight is computed by adding a new weight change to it as follows:

$$\begin{aligned}\text{New Weight Change} = {} & (\text{Learning Rate}) * (\text{Error}) + \\ & (\text{Momentum}) * \\ & (\text{Last Weight Change})\end{aligned}$$

Error is the difference between what an output value is with current network parameters and what the output should be. The Last Weight Change gives both size and direction to the New Weight Change.

The Learning Rate is an adjustable factor that controls the speed of the learning process. The higher the Learning Rate, the faster the learning process. If the rate is too high, however, oscillation in the weight change can impede convergence to an optimal solution set. In contrast, if the Learning Rate is too low, the network can get caught in a local error minimum. This can result in a less than optimal network solution.

In general, network learning can be facilitated by starting with a high Learning Rate. After training has begun, the Learning Rate can be reduced periodically. This speeds learning and reduces the probability that the network solution will settle into a nonoptimal, local error minimum.

The Momentum term is a proportionality constant that has the effect of smoothing Weight Change oscillations dur-

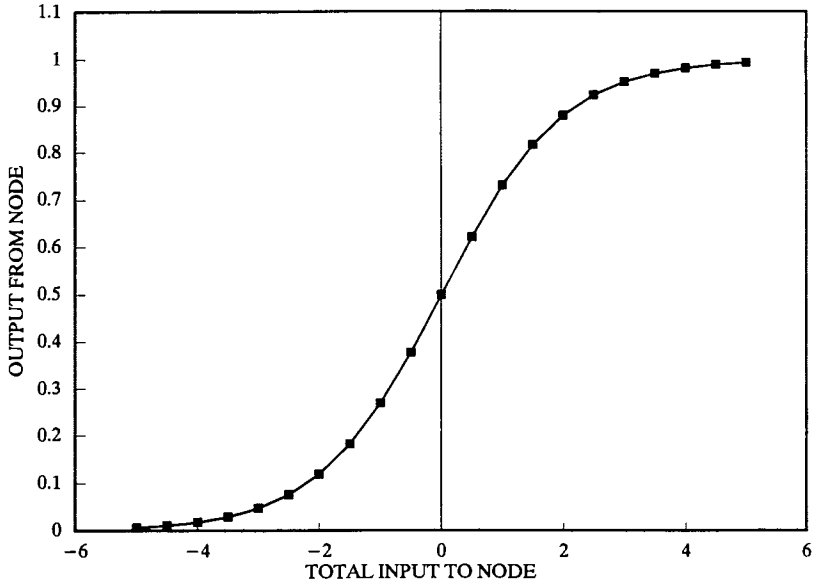

Fig. 4. A sigmoidal transfer function relates the input of a node to its output. The function normalizes output between 0 and 1 and has nonlinear characteristics.

ing the optimization process. Small Momentum values help prevent widely fluctuating New Weight Changes that can impede the network's approach to an optimal solution (8).

Networks can have many optimal solutions. The particular solution reached will depend, in part, on the order in which data pairs are presented to the network. There are two alternatives for the presentation of facts to the network during training. Facts can be presented in the same order each time or presented in random order. Since optimal solution of a trained network may be biased by the order of presentation to the network, it is usually best to present facts to the network in random order during training.

How many and what kind of facts does a network need in order to train properly? Too many facts is generally not a problem. However, the number of facts should not exceed ten times the number of connections (9). Remember that before training, 10–20% of the facts should be randomly set aside to be used for testing the fitted network. Facts should be randomly distributed in the parameter space of the model. Too many facts spatially clumped together or located on the outer fringe of the range of the data set do not allow the model to generalize well.

AN EXAMPLE OF A BPN MODEL: ESTIMATING CREATININE CLEARANCE FROM GENDER, AGE, WEIGHT, AND SERUM CREATININE

To illustrate the modeling process, a BPN was created to estimate creatinine clearance (CL_{cr}) based upon two previously defined relationships adapted from the review by Lott and Hayton (10). The input parameters used were gender, age, weight, and serum creatinine levels:

$$CL_{cr}(\text{males}) = \frac{[140 - \text{Age(years)}] * \text{Weight(kg)}}{72 * \text{Serum Creatinine(mg/dL)}}$$

and

$$CL_{cr}(\text{females}) = \frac{[140 - \text{Age(years)}] * \text{Weight(kg)}}{85 * \text{Serum Creatinine(mg/dL)}}$$

The BPN (Fig. 5) was trained using 200 randomly generated, simulated facts. The input value for gender was defined as having a value of 0 for females or 1 for males. Age ranged from 20 to 90 years. Weight ranged from 40 to 80 kg. Serum creatinine ranged from 0.2 to 2.0 mg/dL. The BPN contained seven hidden nodes. To test the effect of noise on the fitting process, CL_{cr} values were calculated with ±15% random error added. The neural network software used to produce the model was Neuroshell (Ward Systems Group, Inc., Frederick, MD).

After training, a linear regression fit of the fitted model, comparing the actual output of the training facts to the network-estimated output, produced an $R^2 = 0.973$. To test this trained network, outputs for a completely new second set of 200 randomly generated input values (test set) were generated. R^2 between actual and BPN predicted CL_{cr} values was 0.951 (Fig. 6). Figure 7 compares the percentage deviation frequency response for the training set versus the test set. As would be expected, the randomly distributed training set has a (nearly) square distribution due to the even spread of the random error. This square-shaped random distribution is contrasted against the BPN frequency pattern that has forced most of the output errors into the smaller, bell-shaped error range (primarily between ±6%). This demonstrates

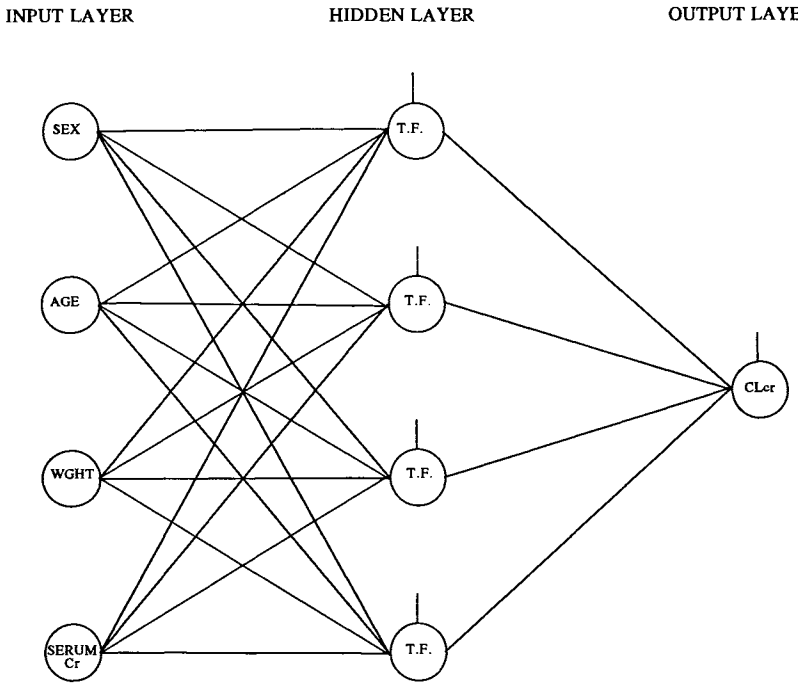

Fig. 5. Network schematic used for the calculation of creatinine clearance (CL_{cr}) from the inputs gender, age, weight, and serum creatinine (Serum Cr). T.F. is the sigmoid transfer function. The network contains seven hidden notes (not all shown).

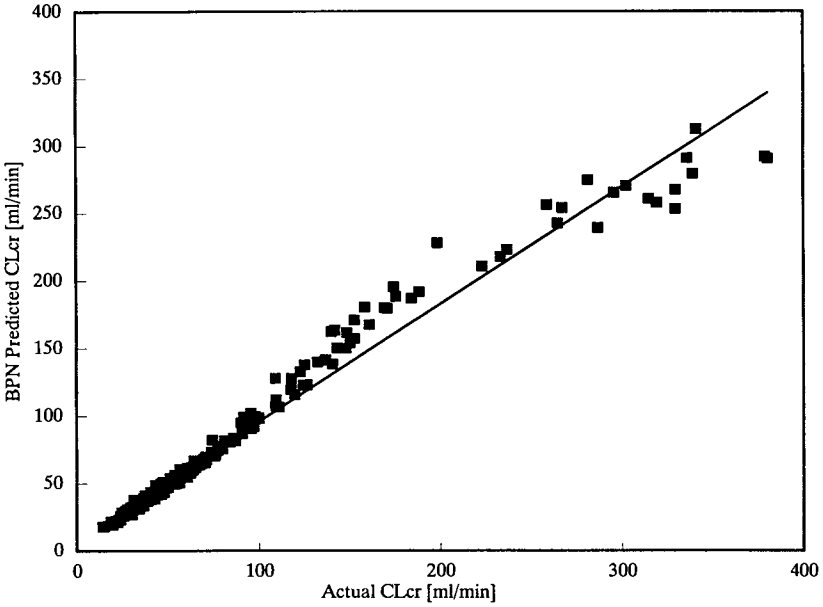

Fig. 6. Regression fit of 200 randomly generated facts used to test the fit of the trained BPN model. $R^2 = 0.951$.

BPN ability to generalize and how it has managed to find a weight connection pattern that attempts to minimize the effect of the random error.

Clearance equations were chosen to model because they are straightforward and generally familiar. The creatinine model also provided an opportunity to incorporate the gender effect into a single model that had been previously described by two separate equations. Based on the training set and test set fits, the BPN clearly had little trouble determining the relationships between the variables. To be fair, however, the BPN model did have 8 bias terms and 35 connection weights to characterize the relationship, so a good fit is not surprising.

GENERAL COMMENTS ABOUT NEURAL NETWORKS

A significant advantage of neural networks is that input variables can be cross-correlated. This is juxtaposed to statistical methods for which cross-correlation can be a significant problem. Neural networks sort out data relationships

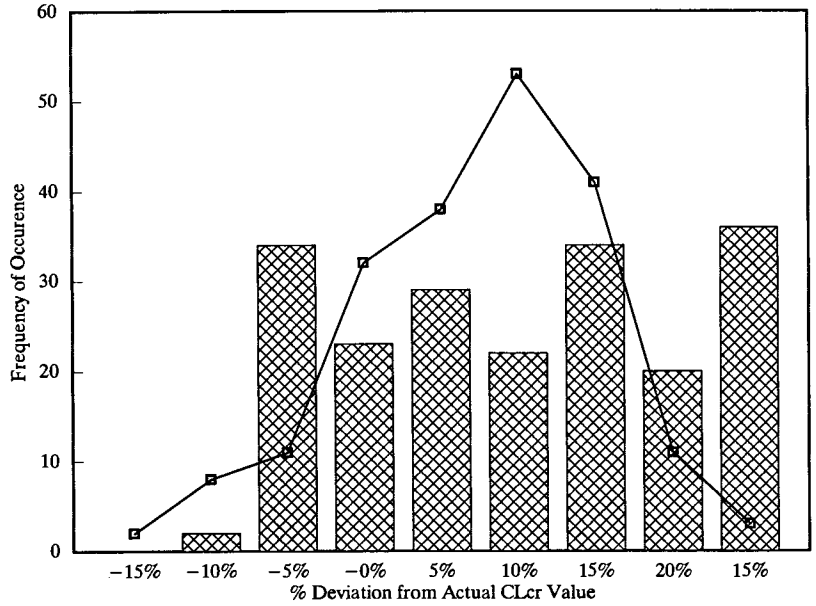

Fig. 7. Comparison of the distribution of the random error (as the percentage deviation from actual) for the training set (bars) versus that of the test set (lines). The BPN model's ability to generalize compresses the deviation error toward the middle of the training set distribution.

and incorporate the most information possible irrespective of cross-correlations between input variables. This facilitates their ability to recognize patterns.

In the sense that neural network computing produces a predictive model, it is an alternative to statistical methods. In the areas of optimization, statistical methods are currently superior to neurocomputing. This is because no methods exist to find optimized input/output solution sets for neural networks. Despite this limitation, neurocomputing competes well with statistical methods in pattern recognition. This is expecially true for problems involving systems containing high levels of noise and variation (11). In fact, Miller *et al.* have described almost 300 neural network applications already in use (12).

Possible areas for the use of neurocomputing in the pharmaceutical industry include elucidation of potential new drug candidates, pharmacokinetic/pharmacodynamic modeling (PK/PD), *in vitro/in vivo* correlations, process control and production, and clinical pharmacokinetics.

In drug discovery, for example, neural networks could be used to determine the classification of a new drug (e.g., stimulant, neuroleptic, antidepressant, or anxiolytic) based upon the EEG effects it produces (Fig. 2) (14). Here, the ability of networks to find obscure patterns in large amounts of noisy data becomes significant.

PK/PD models could be constructed with neural networks without prior consideration of number of compartments. The modeling is strictly model independent. Simulations by the author have demonstrated that hysteresis (the delay in equilibration between the plasma and the effect compartment) can be built into a model to predict temporal drug effects directly from plasma-level data.

Although attempts to relate *in vitro* dissolution data to *in vivo* response have not been very successful, a neural network's ability to provide N-dimensional spatial mapping could lead to new predictive models that were not previously possible.

Pharmaceutical product development and process equipment control can be aided using neural networks (15–17). Process variables, such as fill rates, dwell time, and force of compression parameters in the tableting process, or fermentation parameters are variables that could be monitored and adjusted during production based upon network controllers.

Neural networks could also be useful for describing multiingredient formulation performance by mapping continuous, nonlinear relationships between the amount of ingredients and the physical characteristics of a dosage form (17).

Naturally, the possibility of using networks for clinical pharmacokinetic modeling also represents itself as an obvious option (18).

APPENDIX: GETTING STARTED ON NEUROCOMPUTING

Three textbooks that introduce neurocomputing in practical formats are: *Neural Network PC Tools; A Practical Guide* (14), *Handbook of Neural Computing Applications* (7), and *Neurocomputing* (1).

Of these, the first two texts offer a good place to start because they are less technical and are practically oriented. Indeed, with these texts and other reviews that are available (4,19) the reader can get a workable understanding of neural networks within a relatively short period of time.

With background from these texts, one can proceed to neural network software packages currently available. No computer programming knowledge is required to use these software packages. Two inexpensive PC-based software packages that provide user-friendly interfaces are Neuroshell (Ward Systems Group, Inc., Frederick, MD) and Brainmaker (California Scientific Software, Grass Valley, CA). Both programs are implementations of backpropagation neural networks. Users can begin the modeling process almost immediately using their own data.

REFERENCES

1. R. Hecht-Nielsen. *Neurocomputing,* Addison–Wesley, Reading, MA, 1990.
2. P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in Behavioral Sciences,* Doctoral dissertation, Appl. Math., Harvard University, Cambridge, MA, 1974.
3. D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Vol. 1. Foundations,* MIT Press, Cambridge, MA, 1986.
4. R. P. Lippman. An introduction to computing with neural nets. *IEEE ASSP Mag.* April:4–22 (1987).
5. D. G. Bounds and P. J. Lloyd. A multilayer perceptron network for the diagnosis of low back pain. In *Proc. Second IEEE Int. Conf. Neural Networks,* San Diego, CA, July 24–27, 1988, pp. II-481–II-489.
6. G. Cybenko. Approximations by superpositions of a sigmoidal function. *Math. Control Signals Syst.* 2(4):303–314, (1989).
7. A. J. Maren, C. T. Harston, and R. M. Pap. *Handbook of Neural Computing Applications,* Academic Press, San Diego, CA, 1991, p. 239.
8. R. Hecht-Nielsen. Kolmogorov's mapping neural network existence theorem. In *Proc. First IEEE Int. Joint Conf. Neural Networks,* San Diego, CA, June 21–24, 1987, pp. III-11–III-14.
9. *Neuroshell™ Software Documentation Handbook,* Ward Systems Group, Frederick, MD, 1990.
10. R. S. Lott and W. L. Hayton. Estimation of creatinine clearance from serum creatinine concentration. *Drug Intel. Clin. Pharm.* 12:140–150 (1978).
11. Defense Advanced Research Projects Agency Neural Network Study. Comparison with other technologies. In *Final Report,* AFCEA Int. Press (AIP), Oct. 1987, Feb. 1988, Chap. 19.
12. R. K. Miller, T. C. Walker, and A. M. Ryan. *Neural Net Applications and Products,* SEAI and Graeme, 1990.
13. J. Kuhlman and W. Wingender (eds.), *Dose-Response Relationships of Drugs,* W. Zuckschwerdt Verlag, Munchen, 1990, pp. 157–167.
14. R. C. Eberhart and R. W. Dobbins (eds.), *Neural Network PC Tools: A Practical Guide,* Academic Press, New York, 1990.
15. D. Elrod and R. Trenary. Applications of neural networks in chemistry. 1. Prediction of electrophilic aromatic substitution reactions. *J. Chem. Inf. Comput. Sci.* 30:477–484 (1990).
16. J. Thibault, V. Breusegem, and A. Cheruy. On-line prediction of fermentation variables using neural networks. *Biotechnol. Bioeng.* 36:1041–1048 (1990).
17. A. Hussain, X. Yu, and R. Johnson. Application of neural computing in pharmaceutical product development. *Pharm. Res.* 8:1248–1257 (1991).
18. S. Allerheiligen and T. Luden. Initial evaluation of neural networks in clinical pharmacokinetics. *Pharm. Res.* 7:S-49 (1990).
19. C. Lau and B. Widrow. Neural networks II: Analysis, techniques, and applications. *Proc. IEEE Inst. Elec. Electron Eng.* 78:1547–1549 (1990).