Artificial Neural Networks, Back Propagation, and the Kelley-Bryson Gradient Procedure

Stuart E. Dreyfus*
University of California, Berkeley, Berkeley, California 94720

Introduction

ARTIFICIAL neural networks (sometimes called connectionist, parallel distributed processing, or adaptive networks) are experiencing a dramatic renaissance this decade.

The roots of this subject can be traced to research into perceptrons, led by Frank Rosenblatt, and into adaptive linear filters, spearheaded by Bernard Widrow, in the late 1950s. These early neural-network researchers and their enthusiastic followers equated intelligence with pattern discrimination and association abilities acquired through learning from experience of concrete cases. Then, suddenly, neural-net research became relatively inactive in about 1965 and remained so until the early 1980s. During this interval, research on intelligent systems focused on what has become conventional artificial intelligence, a discipline that defines intelligence as problem solving based on reasoning.[1]

The concurrence of two events probably played a major role in the neural-net resurgence of the 1980s. First, by 1980 it had become increasingly apparent that conventional inference-based artificial intelligence was unable to deal successfully with most practical problems. It appeared that learned pattern discrimination and association abilities, not reasoning, underlay not only common-sense understanding but also most skills. Even the choice of which rules to apply when forced to resort to reasoning, and when to break these rules, seemed to require pattern recognition.[2] Second, a formulational and computational procedure was advanced that seemed to surmount certain technical roadblocks that were recognized but not successfully dealt with by the researchers of the 1950s and 1960s. This procedure is called back propagation (BP) (of errors) and is the subject of this Note.

To fully appreciate BP, we must first briefly examine some groundbreaking work done during the late 1950s. At Cornell, Frank Rosenblatt designed various neurally inspired learning devices and simulated them on a digital computer. He called these designs "perceptrons" to emphasize their perceptive, rather than logical, abilities. The aim of many of his devices was to learn through example to distinguish whether an input was a member of one class of inputs, called class A, or of a different class, called class B, by being presented with examples of members of each class together with the correct classification. The class members, any finite number being allowed, were represented by $(n-1)$ vectors where x_i^c denotes the ith element of the cth such vector. Given a vector, the simplest perceptron would compute

$$\sum_{i=1}^{n-1} w_i x_i^c - t$$

where w_i were viewed as weights, adjustable during learning and similar in role to synapses in the brain, applied to the input elements x_i^c, and t, also adjustable, was called a threshold. Hence the weighted sum of the input's elements was compared to a threshold. Letting net^c denote the amount by which the weighted input exceeds the threshold t (where net^c can be negative), the output unit (artificial neuron) would output a 1 if net^c exceeded 0, and a -1 otherwise. One can describe this threshold linear unit mathematically by the input-output relation for case c

$$o^c = f(\text{net}^c) = f\left(\sum_{i=1}^{n-1} w_i x_i^c - t\right) \tag{1}$$

where the function f takes on value 1 for strictly positive argument and -1 otherwise.

To avoid always distinguishing the threshold variable from the weights in formulas, it is conventional to consider the input vectors to be of dimension n (rather than $n-1$) with the nth component always 1. Then $-t$ is written as w_n and Eq. (1) becomes simply

$$o^c = f\left(\sum_{i=1}^{n} w_i x_i^c\right)$$

We shall adopt this convention in what follows.

Suppose that all members of class A of inputs are associated with a desired output d of 1 and of class B with an output of -1. Then, weights (the nth of which is really the negative of the threshold) are sought during learning that correctly give the desired output for all training cases.

Rosenblatt, with embellishments added by others, proved the perceptron convergence theorem, which asserts that if there exist any weights that yield the desired output for all training cases, one such set will be found in a finite number of steps if one starts with all weights at zero and iterates over and over through the training set, and at each presentation of an input vector x^c w is modified by the rule

$$w_i(\text{new}) = w_i(\text{old}) + \frac{\eta}{2}(d^c - o^c)x_i^c \tag{2}$$

where η is any positive constant.

A simple perceptron is, unfortunately, able to correctly classify cases into two classes only if the cases are linearly separable. This led in the early 1960s to a search for a way to accomplish classification using multilayer devices that would work even when the cases were not linearly separable. It was found that a feedforward perceptron with two adjustable layers of weights, using on layer 1 threshold linear units acting on the weighted linear combination of layer-zero input data and another such unit on layer 2 acting on the weighted linear combination of layer-1 outputs, is able to classify correctly any finite number of cases if there are a sufficient number of "hidden units" in the layer 1 of units mediating between the input vector and the output unit. A multilayer network with more than two layers can sometimes do the same job with fewer hidden units or with hidden units that can be interpreted as higher-order feature detectors. Therefore, in what follows, we shall consider N-layer devices for arbitrary N. Unfortunately, for fewer hidden nodes than cases to be learned, no even moderately efficient algorithm for systematically discovering connection weights to do the job could be found. BP represents significant progress toward resolution of this difficulty.

Multilayer Network Learning as a Discrete-Stage Optimal Control Problem

The key idea is to turn the problem of finding connection weights satisfying the classification requirements into a nonlinear, discrete-stage optimal control problem with connection weights as control variables. Since the standard control problem assumes differentiable dynamical equations, the threshold linear unit will no longer do. Instead, researchers addressing this problem recently began to propose using a differentiable

Received July 5, 1989; revision received Oct. 7, 1989.

*Professor, Department of Industrial Engineering and Operations Research.

nonlinear function rather than a threshold function to map $\text{net}_i^c(k)$ (the net input into unit i of layer k for case c) into its output $x_i^c(k)$. A commonly used function, when the output is restricted to the range -1 to 1, is

$$x_i^c(k) = f\left[\text{net}_i^c(k)\right] = \tanh\left[\text{net}_i^c(k)\right] = \frac{1-e^{-\text{net}_i^c(k)}}{1+e^{-\text{net}_i^c(k)}} \tag{3}$$

$$\text{net}_i^c(k) = \sum_{j=1}^{n} w_{ij}(k-1)x_j^c(k-1) \tag{4}$$

where $w_{ij}(k-1)$ is the weight of the connection to unit i of layer k from unit j of layer $k-1$. In what follows we shall restrict ourselves to the particular form of Eq. (4), but we shall make no assumption about f except differentiability. The preceding commonly used tanh function f has the property that the output of a unit approaches 1 only as its input approaches infinity, and -1 only as its input approaches minus infinity. Recognizing that these formulas can achieve outputs of 1 or -1 only with infinite weights, the classification goal is modified. We shall consider the goal to be to find weights that cause the output of the output unit to equal 0.9 when the input is from class A and to equal -0.9 when the input is from class B, where A and B are the two input classes that the network is to learn to distinguish.

We can now state the multistage optimal control problem that the classification task requires us to solve. First, we define the following:

N = number of layers of adjustable weights in the network
$w_{ij}(k)$ = weight on connection from unit j of layer k to unit i of layer $k+1$, $k=0,\ldots,N-1$
n_k = number of units in layer k [including unit n_k, which has output of 1 on each layer; hence $w_{in_k}(k)$ is the negative of the threshold of unit i of layer $k+1$]; for classification into two classes $n_N = 1$
$x_i^c(k)$ = net output of unit i of layer k for case c; $x_{n_k}^c = 1$ for $k=0,\ldots,N-1$
d^c = desired output (0.9 or -0.9) of unit 1 of layer N for case c
C = number of cases
$x_i^c(0)$ = ith element of the input vector for case c

Our control problem is of the Mayer type and has terminal criterion

$$J = \frac{1}{2}\sum_{c=1}^{C}[d^c - x_1^c(N)]^2 \tag{5}$$

which measures the squared error of the deviation of the actual output from its desired value summed over all cases. The dynamical relations that map the input (the state at stage 0) for case c into the output (the state at stage N) are, by Eqs. (3) and (4),

$$x_i^c(k+1) = f\left[\text{net}_i^c(k+1)\right] = f\left[\sum_{j=1}^{n_k} w_{ij}(k)x_j^c(k)\right] \qquad k=0,\ldots,N-1 \tag{6}$$

The initial condition $x_i^c(0)$ is given, with $x_{n_0}^c(0) = 1$ for all c.

To render this a completely standard control problem with one initial state vector, rather than one input vector for each case to be learned, we simply concatenate the C state vectors, each of length n_k at stage k, into one vector of length Cn_k and duplicate the network for each case. Figure 1 shows the resulting two-layer network for four cases with three inputs (the last of which is always 1) per case.

Solution by Back Propagation – Kelley-Bryson Gradient Method

The gradient-solution procedure for optimal control problems was developed by Kelley[3] in 1960 and, independently, by Bryson[4] at about the same time. These airplane and rocket trajectory researchers generally considered continuous-time dynamics and more complex terminal conditions than we have here. Bryson did, however, present a gradient solution of a multistage problem in Ref. 5 and Bryson and Ho explicitly gave in their 1969 book[6] gradient formulas for exactly the multistage, free-terminal-state problem we are considering. Kelley used adjoint equations and Green's theorem in his derivation, and Bryson used Lagrange multipliers. Dreyfus,[7] in 1962, used a simple, new recursive derivation based on the chain rule of differentiation to obtain results of known Kelley-Bryson type and dealt explicitly with the optimal control problem in its discrete-stage form. Though neural-net researchers have come to recognize that multistage feedforward nets fit into the optimal control theory mold and that BP is a gradient procedure, proper credit for the BP method of solution has not been accorded to Kelley and Bryson. Often the 1974 doctoral dissertation of Werbos[8] is cited as the earliest reference. Werbos independently developed a derivation similar to that of Dreyfus, and considered feedforward dynamics more general than the multistage model usually used in optimal control problems and neural network design that we are considering here.

The discrete-stage form of the Kelley-Bryson formulas, when applied to the case of N adjustable-weight layers and C cases, yields

$$\bar{\delta}_1^c(N) = x_1^c(N) - d^c \qquad c=1,\ldots,C \tag{7}$$

$$\bar{\delta}_i^c(k) = \sum_{j=1}^{n_{k+1}} w_{ji}(k)f_j^{c\prime}(k+1)\bar{\delta}_j^c(k+1) \quad i=1,\ldots,n_k-1; \quad c=1,\ldots,C; \quad k=0,\ldots,N-1 \tag{8}$$

$$\nabla J_{w_{ij}(k)} = \sum_{c=1}^{C} x_j^c(k)f_i^{c\prime}(k+1)\bar{\delta}_i^c(k+1) \quad i=1,\ldots,n_{k+1}-1; \quad j=1,\ldots,n_k; \quad k=0,\ldots,N-1 \tag{9}$$

From the Dreyfus derivation, it is obvious that the symbol $\bar{\delta}_i^c(k)$ represents $\partial E_k/\partial x_i^c(k)$ where E_k is defined as the terminal cost, given the weights, and is viewed as a function of the initial condition vector at stage k, $x_i^c(k)$, and the weights at stage k, $w_{ij}(k)$. The gradient $\nabla J_{w_{ij}(k)}$ is $\partial E_k/\partial w_{ij}(k)$.

Examining these results we see that we can decompose the computation and perform it on a case-by-case basis, using

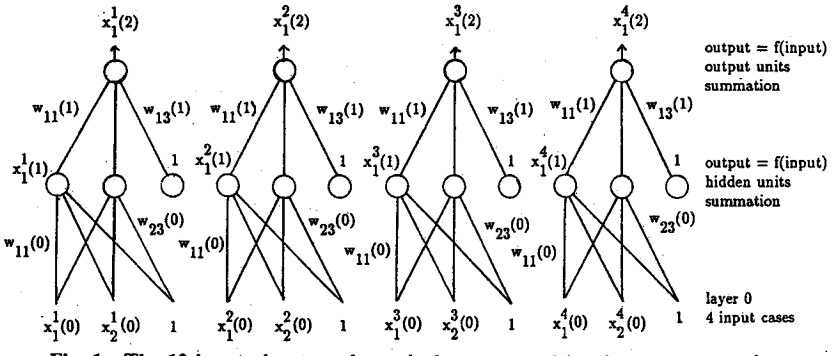

Fig. 1 The 12-input, 4-output dynamical system resulting from concatenation.

Eqs. (7) and (8) to compute $\bar{\delta}_i^c(k)$ and one term of the summation on the right-hand side of Eq. (9) for case c, and then summing over the cases to obtain the gradient given by Eq. (9).

Once the first-order effects on the terminal cost of weight changes have been determined, a variety of procedures have been proposed in the control and neural-net literature for actually modifying the weights prior to a repetition of the preceding procedure.

The best-known BP results[9] look different from ours but are equivalent. They are stated in terms of quantities $\delta_i^c(k)$ [instead of our $\bar{\delta}_i^c(k)$], and these quantities are interpretable as the negative of the partial derivative of the criterion value with respect to the *net input* to unit i on layer k for case c rather than the positive partial derivative with respect to the output.

Sometimes classification of cases into more than two classes, or the mapping of input vectors into associated output *vectors*, is desired. Then more than one unit is located on the output level for each case. In both instances only a minor modification of the preceding results is required, with the squared-error criterion now involving a summation over all output units for each case.

In a strict mathematical sense, BP has not solved the problem of efficiently determining weights in multilayer networks such that a net will produce stipulated results for a training set of cases. Like any gradient procedure applied to nonlinear and generally nonconvex problems, convergence to local minima (or other stationary points) with nonzero error is possible, even when weights yielding zero error exist. A mathematician would demand a procedure known to yield a solution if one exists [which is the case for the one-layer perceptron using the perceptron learning rule of Eq. (2)] before he or she would consider the problem solved. Additionally, the error function viewed as a function of the weights sometimes has many saddle points that, even if the gradient procedure never exactly reaches one, cause the procedure to take many thousands of steps before convergence on a local minimum. This possibility renders the procedure's efficiency poor and also, thus far, unanalyzable.

Conclusion

Once all of the cases to be learned in a neural-net mapping problem are concatenated into one large network with a vector output, one component for each case, a standard discrete-time optimal-control problem results. The Kelley-Bryson gradient formulas for such problems have been rediscovered by neural-network researchers and termed back propagation. The recursive derivation of these formulas using the chain rule, commonly seen in the neural-network literature, was first used for optimal-control problems by Dreyfus.

References

[1]Dreyfus, H., and Dreyfus, S., "Making a Mind vs Modeling the Brain: Artificial Intelligence Back at a Branchpoint," *Daedalus*, Winter 1988, pp. 15–43.

[2]Dreyfus, H., and Dreyfus, S., *Mind over Machine: The Power of Human Intuition and Expertise in the Era of the Computer*, Free Press, New York, 1988.

[3]Kelley, H. J., "Gradient Theory of Optimal Flight Paths," *ARS Journal*, Vol. 30, No. 10, 1960, pp. 947–954.

[4]Bryson, A. E., and Denham, W., "A Steepest-Ascent Method for Solving Optimum Programming Problems," *Journal of Applied Mechanics*, Vol. 29, No. 2, 1962, pp. 247–257; see also Bryson, A. E., et al., "Determination of the Lift or Drag Program that Minimizes Reentry Heating with Acceleration or Range Constraints Using a Steepest Descent Computation Procedure," Inst. of Aeronautical Sciences, Paper 61-6, 1961.

[5]Bryson, A. E., "A Gradient Method for Optimizing Multi-Stage Allocation Processes," *Proceedings of the Harvard University Symposium on Digital Computers and Their Applications*, April 1961.

[6]Bryson, A. E., and Ho, Y. C., *Applied Optimal Control*, Blaisdell, Waltham, MA, 1969, pp. 43–45.

[7] Dreyfus, S., "The Numerical Solution of Variational Problems," *Mathematical Analysis and Applications*, Vol. 5, No. 1, 1962, pp. 30–45.

[8]Werbos, P., "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Ph.D. Dissertation, Committee on Applied Mathematics, Harvard Univ., Cambridge, MA, 1974.

[9]Rumelhart, D., Hinton, G., and Williams, R., *Parallel Distributed Processing*, Vol. 1, edited by D. Rumelhart, J. McClelland, and the PDP Research Group, MIT Press, Cambridge, MA, 1986, Chap. 8.

Book Announcements

VUKOBRATOVIC, M., BOROVAC, B., SURLA, S., and STOKIC, D., *Biped Locomotion: Dynamics, Stability, Control and Application,* Springer-Verlag, Berlin, 1990, 349 pages.

Purpose: This volume is intended for researchers interested in the study of biped gait and its stabilization.

Contents: Dynamics of biped locomotion; synthesis of nonlinear dynamics; control and stability; realizations of anthropomorphic mechanics; appendices.

MEIROVITCH, L., *Dynamics and Control of Structures,* Wiley, New York, 1990, 425 pages.

Purpose: Various disciplines involved in the control of structures, namely, analytical mechanics, structural dynamics, and control theory, are represented in this book. The coverage of each discipline is sufficiently detailed to provide a broad picture of the field of structure control.

Contents: Newtonian mechanics; principles of analytical mechanics; concepts from linear systems theory; lumped-parameter structures; control of lumped-parameter systems, classical and modern approaches; distributed-parameter structures, exact and approximate methods; control of distributed structures; review of literature on structural control.

LENT, B., *Dataflow Architecture for Machine Control,* Research Studies Press, Somerset, U.K., 1989, 315 pages.

Purpose: This book presents an approach to achieve better price/performance relations for machine control based on the concept of dataflow driven systems.

Contents: Definitions and taxonomy of computer architectures for machine embedded control systems; computer architectures deployed in machine embedded control systems; performance analysis of computer architectures deployed in machine embedded control systems; computer operation in terms of tokens; operational principles for OR tokens; hardware structure supporting the OR dataflow operational principle; OR dataflow operating system; programmability; machine control system based on OR dataflow computer architecture.

SCHIEHLEN, W., (Ed.), *Multibody Systems Handbook,* Springer-Verlag, Berlin, 1990, 432 pages.

Purpose: This handbook consists of a collection of programs and software on multibody formalisms contributed by numerous international authors.

Contents: Overview; test examples; description of codes; contributors and distributors.