

BOOK REVIEW

Code recognition and set selection with neural networks, by Clark Jeffries. Birkhäuser, Boston, 1991, 166 pp., \$49.50. ISBN 0-8176-3585-8

Neural networks are computational schemes modeled vaguely on certain observed or conjectured features of biological neurons. Think of a finite set of processing units arranged as the vertices of a directed graph. An arrow in the graph from unit j to unit i means that j sends its output to i . Each unit performs a simple computation on the inputs it receives, its output being the result of that computation. The computation may be deterministic or stochastic, running in discrete or continuous time. The networks in this book run in continuous time, with outputs of units determined by differential equations.

There are two kinds of neural networks: the actual nervous systems of organisms, studied by biologists, and the computational models more accurately called “artificial neural networks”, investigated by engineers, psychologists, physicists, and others, including a few mathematicians. Up to now the biological networks have proved too complex to yield to mathematical analysis, and the mathematical analysis of artificial networks is in its infancy. A growing body of theory is developing, however, and challenging mathematical problems are emerging. *Code recognition* is a noteworthy contribution to this effort.

The first neural computational schemes were the McCulloch-Pitts nets of artificial neurons in the 1940s ([15], reprinted in *Neurocomputing* [1]). Intense interest was generated in the late 1950s by learning theorems found by Rosenblatt [18] for his “Perceptrons”; at the same time Widrow invented similar networks called “Adalines” (see Widrow [23]; Widrow and Hoff [22], reprinted in *Neurocomputing*). These nets could learn to perform certain boolean maps by simple algorithms which adjusted parameters called *weights* to reduce error. Not every map could be performed by a given perceptron, but Rosenblatt’s Perceptron Learning Algorithm will always find a set of weights realizing a given map if such weights exist.

The elegant 1969 book *Perceptrons* by Minsky and Papert [17] analyzed the simplest perceptrons, judged them to be very limited, and concluded that the new field of artificial intelligence should forget them. This conclusion, which even the authors now admit was overly pessimistic, all but killed interest in neural computation. Kept alive by lonely pioneers such as S. Amari, E. Canaiello, J. Cowan, S. Grossberg, and others, the field of neural networks received strong new stimuli in the 1980s from the independent rediscovery by several people of a practical method of estimating optimal weights for specific tasks through gradient descent on error functions (Rumelhart and McClelland [19].) Physicist Hopfield ([10, 11],

reprinted in *Neurocomputing*) aroused great excitement with his design of neural networks and learning algorithms which were able to find reasonable solutions to combinatorial problems such as the Traveling Salesman Problem.

Today this field is rapidly expanding, with a host of new professional organizations, journals, books, academic units, and annual conferences. Neural networks are being used for pattern recognition and combinatorial optimization; for making bankers' decisions on loan risks; diagnosing headaches; detecting explosives in airplane luggage; decoding sonar signals; composing music in the style of Mozart or Gershwin; predicting EEGs, currency exchange rates, sunspots, the Mackey-Glass equation, and other time series; controlling and programming robots; and any number of other tasks. The big hardware breakthroughs—a chip incorporating easily programmable weights, holographic optical computers—are always just around the corner but remain elusive (see Mead [16]). Currently, neural nets are simulated on standard computers, precluding exploitation of the inherent massive parallelism in network computations and effectively limiting the size of networks.

Neural networks have been welcomed by some psychologists, linguists, cognitive scientists, and philosophers as providing a model of the mind which seems more plausible than the currently popular view of the mind as a computer program or a formal logical system. Such discussions are occasionally refreshingly polemical.

The practical objective of neural network research is to learn how to design networks for performing a given task. A successfully designed network is used as an expert system. Traditional expert systems incorporate explicit rules given by the designer of the system. Neural networks, on the other hand, learn from experience: Upon being presented with typical examples of the task and their solutions, a learning algorithm adjusts weights and perhaps other parameters to improve performance. The designer of the network selects the learning algorithm but does not know what the best rules are—if she knew them, she would not need a network. After the network has been successfully trained it may be possible to examine the weights and infer rules that the network seems to be obeying. A spectacular example is the network “NetTalk” [20], which learns to convert printed English into comprehensible speech. No explicit rules of pronunciation are built into the network—the only thing the network “knows” is how to adjust its weights to improve performance. The same program would work equally well for Spanish or German (modulo umlauts).

A much admired virtue of networks is their robustness: Changing a few weights or deleting a few units will often not seriously degrade the performance. On the other hand, while networks excel at pattern recognition, some critics claim they are not suited for cognitive tasks requiring long-term memory and convoluted chains of symbolic reasoning. This is a controversial issue requiring much research.

Success is typically defined as some error function having small value, and the designer tries to choose weights to minimize this function. This is often done by some version of gradient descent, considering the error function as a function on the weight space. There are not only the usual problems with gradient descent—slow convergence and false local minima—but for recurrent nets (for which the graph has directed loops) the error function may be discontinuous. Another difficulty is that the only way to compute the error function is by running the network with many different inputs; there is no explicit formula from which to work. A great deal of research has gone into the problem of finding weights for feed-forward nets (no loops in the directed graph) and into the more general problem of adapting the

network architecture to particular problems.

For recurrent nets logically prior to the problem of choosing weights there is the job of figuring out the dynamics (which are trivial for feed-forward nets). Usually we want most trajectories to approach one of several stable equilibrium states or perhaps a stable limit cycle. This limit set *attractor* can then be considered the system's output, the input being the initial state of the trajectory.

While chaotic systems provide a rich—indeed, seemingly unmanageable—repertoire of dynamical behavior, their use for neural networks is problematical. What would we do with chaotic output? If our net feeds into another net, this output becomes chaotic input—how could a net handle that? Yet many observations of nervous activity reveal apparently chaotic dynamics, and some biologists believe it has biological advantages (see Skarda and Freeman [21]).

The networks studied in this book are mathematical models of analog computing devices. They are dynamical systems expressed by systems of differential equations having the specific algebraic forms given below. Such systems are viewed as input–output devices: the input is the initial state, the output is the attractor to which the trajectory converges. The problem is to design the system in order to obtain a desired input–output map.

In the author's words:

Much of this book is devoted to a neural network method for recognizing binary strings which have been corrupted by noise in transmission . . . It appears that an electronic analog of the mathematical error-correcting decoder would correct errors faster and more reliably than conventional algebraic decoders.

Jeffries's book exhibits and analyzes explicit networks which find reasonable solutions to various small-scale problems in optimization, error correction, and pattern recognition. While these problems involve only finite sets, the neural nets that solve them use not discrete mathematics but differential equations. One of the most interesting discoveries in neural net research has been that continuous dynamical systems can be effectively used to solve purely discrete problems, such as the Traveling Salesman Problem (see Hopfield [10].)¹

The systems studied by Jeffries have the form:

$$(1) \quad \frac{dx_i}{dt} = -k_i x_i + p_i(g_1(x_1, \dots, x_n), \dots, g_n(x_1, \dots, x_n)), \quad i = 1, \dots, n.$$

This describes the dynamics of a network of n units. The *activation state* of unit i is $x_i \in \mathbf{R}$, which is converted to the unit's *output signal* by the nondecreasing *gain function* $g_i: \mathbf{R}^n \rightarrow \mathbf{R}$. The signals coming into unit i interact via the functions p_i , which are real-valued polynomials in n variables. The *decay rates* k_i are positive constants.

Networks will recognize these equations as defining *higher-order* networks, the order here corresponding to the highest-order monomial appearing among the p_i . Such networks have not been studied much, despite their demonstrated computational power (see Chen et al. [3], Giles and Maxwell [5]). Even first-order networks—with linear p_i —can exhibit quite chaotic behavior in simulations, and their dynamics is poorly understood.

¹This should be pondered by the misguided souls who propose to reform mathematics teaching by eliminating calculus.

If a monomial $X_{j_1} \cdots X_{j_k}$ appears in p_i with nonzero coefficient $W = W_{i,j_1,\dots,j_k}$, this means that the signals output from units j_1, \dots, j_k combine multiplicatively before entering unit i . This combined signal tends to excite unit i (in the sense of increasing the activation level) if the weight W is positive and to inhibit unit i if W is negative.

A famous class of networks comprise the *additive* networks:

$$(2) \quad \frac{dx_i}{dt} = -k_i x_i + \sum_{j=1}^n W_{ij} g_j(x_j) + I_i, \quad i = 1, \dots, n.$$

Here the polynomials p_i are linear. The I_i are constant *external inputs*. Such systems as well as their discrete-time and stochastic analogs have been studied by many authors. Equations of this kind are sometimes “derived” from suggestive biological or electrical analogies. A major challenge, however, is to convincingly obtain equations of network dynamics from biological principles.

In most networks the gain functions $g_i(x_i)$ are either linear or, as in this book, sigmoidal: bounded with nonnegative derivatives tending to zero as $|x_i| \rightarrow \infty$. The maximum derivative is an important parameter called the *gain*.

Usually the gain functions are Lipschitz continuous, guaranteeing the usual theorems of existence and uniqueness of solutions. But in Theorem 3.3 a slightly discontinuous gain function appears, which is not unusual in network models; Heaviside functions are common. The theory of such discontinuous dynamical systems remains to be worked out, however (see Kidder [12], Cowan [4]).

It is not the activations x_i but the outputs $g_i(x_i)$ that are of interest in applications. Typically the g_i are chosen so that outputs are close to 1 for $x_i > \varepsilon_i$ and to 0 for $x_i < -\varepsilon_i$; sometimes -1 is used instead of 0. If this holds for all i , I call such a state vector x *saturated*. The output vector at a saturated state thus approximates a sequence of n zeroes and ones, which can be taken as the coding for some pattern of interest.

Such a code can be interpreted as representing subsets of the set $\{1, \dots, n\}$. Suppose we need to find a subset having some desired combinatorial property. We could try to do this by designing a network with the following properties:

- (a) Almost every trajectory converges to a stable equilibrium.
- (b) Many stable equilibria are saturated.
- (c) The sets of the desired type correspond bijectively to the output vectors at saturated stable equilibria.

In Chapter 1 Jeffries applies just this method to the *set selection problem*: Given a finite list of finite sets, find a set that intersects each of them in exactly one element. For each such problem he writes down a neural net having the properties listed above. Here there is no learning; the weights are determined from the particular set selection problem. The gain functions have the simple form of continuous, nondecreasing ramp functions: constant outside a small interval and linear inside the interval. This makes the analysis of the dynamics near stable equilibria rather simple.

Several set selection problems are analyzed. These also illustrate the unfortunate but common difficulty that there may be stable equilibria that are not saturated (having some components in the interval $[-\varepsilon_i, \varepsilon_i]$) and which are thus not meaningful for the set selection problem. Several methods of ameliorating this problem are

discussed. The chapter closes with exercises and solutions, including the new (to me) idea of converting the Euler numerical approximation method for equation (1), for a set selection problem, to a spread-sheet algorithm. There are several other spread-sheet algorithms in the exercises.

For most recurrent networks based on systems of differential equations one wants assurance that every trajectory converges to some equilibrium. Even low-dimensional examples of (1) can have cycles (nonconstant periodic orbits). Chapter 2 applies previous work of the author to this problem. Jeffries associates to certain systems of type (1) a combinatorial scheme called a *hypergraph*, which is a finite graph with signed edges and some labeled vertices. Under certain conditions on the hypergraph Jeffries constructs a Liapunov function for system (1), i.e., a continuous real-valued function on the state space which decreases on nonconstant trajectories and which is bounded below. The existence of such a function precludes chaotic dynamics or cycles: every trajectory must approach the equilibrium set. And for the special dynamics of (1) Jeffries shows every trajectory must converge.

A number of authors have studied networks based on equation (2) as *content addressable memories*, and Jeffries uses the more complex system (1) for the same purpose. In equation (1) we now assume the gain functions $g_i(s)$ limit at ± 1 as $s \rightarrow \pm 1$ respectively. Suppose we have vectors $X^\alpha \in \mathbf{R}^n$, $\alpha = 1, \dots, m$, with all components being ± 1 . We call these the *memories*. We want the dynamics of (1) to have the X^α as asymptotically stable equilibria—briefly, *attractors*. In that case we can access a memory X^α by choosing as initial value v for the dynamics any vector in the basin of attraction of X^α . Such an initial value may differ considerably from the correct memory. For example, X^α might be a coding of a photograph, while v may be a coding for the upper left-hand corner or, alternatively, of a different photograph of the same scene. Striking examples of this use of networks may be found in Kohonen's book [13]. The memories in Jeffries's book are much simpler.

The major problem in designing this kind of network (and most other kinds) is to choose the weights to achieve the required dynamics. There are many different algorithms for this, each with its own advantages and disadvantages.

In subsequent chapters Jeffries builds on the dynamical, algebraic, and combinatorial results of the first two chapters to construct neural nets for several problems: content addressable memory, code recognition, digital communication, storing memories as limit cycles, and certain operations research problems. In the final section he presents the quadratic assignment problem with admirable candor as one which "simple neural network models seem powerless to solve well".

It is common in neural network research to present a new method, test it by simulation on a hard problem, and compare the results to previous methods, hopefully demonstrating the superiority of the new methods. That is not Jeffries's goal at all. Rather, he proves theorems (rare in this subject!) about a large class of networks and illustrates their use as clearly as possible on very simple problems. Having read the book and worked through the problems at the end of each chapter (answers are supplied), the reader will be able to test the methods on her own problems.

The literature on neural networks is growing rapidly. This is one of the more mathematically interesting books. Deliberately limited in scope, it makes clear the use of networks in solving a number of combinatorial problems.

The bibliography below lists several recent books on neural nets having different perspectives. The two books [6, 7] by Grossberg contain many network models of various biological functions. Many of Grossberg's basic ideas are clearly expounded

in Levine [14]. Hertz et al. [9] present an approach, common among physicists, which exploits a fruitful analogy between certain nets and spin-glasses. Hecht-Nielsen [8] reviews many engineering approaches to practical network construction. The two volumes coedited by Anderson [1, 2] are rich collections of papers on neural networks going back to William James! Kohonen's book [13] is a mathematically oriented account of several memory models. The Rumelhart and McClelland volumes [19] are oriented toward cognitive models.

REFERENCES

1. J. Anderson and E. Rosenfeld (eds.), *Neurocomputing: Foundations of research*, MIT Press, Cambridge, MA, 1988.
2. J. Anderson, R. Rosenfeld, and A. Pellionisz (eds.), *Neurocomputing 2: Directions for research*, MIT Press, Cambridge, MA, 1990.
3. H. H. Chen, Y. C. Lee, G. Z. Sun, H. Y. Lee, T. Maxwell, and C. L. Giles, *High order correlation model for associative memory*, Neural Networks for Computing AIP-Conf. Proc., vol. 151, Amer. Inst. Phys., New York, 1986, p. 86.
4. Stuart Cowan, *Dynamical systems arising from game theory*, Ph.D. thesis, Univ. of California at Berkeley, 1993.
5. C. Giles and T. Maxwell, *Learning, invariance, and generalization in higher-order neural networks*, Appl. Optics **26** (1987), 4972–4978.
6. S. Grossberg, *The adaptive brain*, Advances in Psychology, North-Holland, New York, 1987.
7. S. Grossberg and M. Kuperstein, *Neural dynamics of adaptive sensory-motor control*, Pergamon Press, New York, 1989.
8. R. Hecht-Nielsen, *Neurocomputing*, Addison-Wesley, Reading, MA, 1990.
9. J. Hertz, A. Krogh, and G. Palmer, *Introduction to the theory of neural computation*, Santa Fe Institute studies in the sciences of complexity, Lecture notes, vol. 1, Addison-Wesley, Reading, MA, 1991.
10. J. Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, Proc. Nat. Acad. Sci U.S.A. **79** (1982), 2554–2558.
11. ———, *Neurons with graded responses have collective computational properties like those of two-state neurons*, Nat. Acad. Sci. U.S.A. **81** (1984), 3988–3992.
12. J. Kidder, *A theory of faulty vector fields*, Ph.D. thesis, Univ. of California at Berkeley, 1992.
13. T. Kohonen, *Content-addressable memories*, Springer-Verlag, New York, 1980.
14. D. Levine, *Introduction to neural and cognitive modeling*, L. Erlbaum Associates, Hillsdale, NJ, 1991.
15. W. McCulloch and W. Pitts, *A logical calculus of ideas immanent in nervous activity*, Bull. Math. Biophys. **5** (1943), 115–133.
16. C. Mead, *Analog VLSI and neural systems*, Addison-Wesley, Reading, MA, 1989.
17. M. Minsky and S. Papert, *Perceptrons; an introduction to computational geometry*, expanded ed., MIT Press, Cambridge, MA, 1988.
18. F. Rosenblatt, *Principles of neurodynamics; perceptrons and the theory of brain mechanisms*, Spartan Books, Washington, DC, 1962.
19. D. Rumelhart, J. McClelland, and the PDP research group, *Parallel distributed processing: explorations in the microstructure of cognition*, vols. 1, 2, MIT Press, Cambridge, MA, 1986.
20. T. Sejnowski and C. Rosenberg, *Parallel networks that learn to pronounce English text*, Complex Systems **1** (1987), 145–168.
21. A. Skarda and W. J. Freeman, *How brains make chaos in order to make sense of the world*, Behavioral and Brain Science **10** (1987), 161–195.
22. B. Widrow, *Generalization and information storage in networks of adaline “neurons”*, Self-Organizing Systems 1962 (M. Yovits, G. Jacobi, and G. Goldstein, eds.), Spartan Books, Washington, DC, 1962, pp. 435–461.
23. B. Widrow and M. Hoff, *Adaptive switching circuits*, Neurocomputing (J. Anderson and E. Rosenfeld, eds.), MIT Press, Cambridge, MA, 1988.

MORRIS W. HIRSCH
UNIVERSITY OF CALIFORNIA AT BERKELEY
E-mail address: hirsch@math.berkeley.edu