Book Reviews

Communication Theory: Transmission of Waveforms and Digital Information—D. J. Sakrison (New York: Wiley, 1968, 369 pp., $12.95).

The author of an introductory book has an obligation to do more than just present a specified set of material to the student. He must develop a rapport with the reader and give him the feeling that this is "where the action is." Sakrison has done an admirable job in this introductory survey of modulation and reception systems covering AM, FM, PM, PPM, PCM, and phase-locked loop performance in the presence of additive noise. The material in the book can be easily read and understood by engineering students at the senior level. Forming an integral part of the text are the exercises, which are placed with the pertinent material rather than at the end of the chapter. These exercises are chosen to be easily solved if the material is well understood. By looking at specific systems, important concepts such as trading bandwidth for signal-to-noise ratio, the threshold effect, and the capability of digital coding arise in an intelligible and intuitive manner. Although most of the material is available in much greater detail in other books, it could not be read with clarity by students uninitiated in communication theory.

Since the book is intended to be self-contained, the first part consists of a terse and highly specific development of Fourier transforms, Laplace transforms, and probability theory. Some of the effort required in the development from first principles seems unnecessary since, for a student to readily comprehend the book, his background should already include most of this material. The second part of the book depends heavily upon the works of Davenport and Root for the treatment of random processes, Abramson for FM, Wozencraft and Jacobs for digital data and nonlinear modulation, and Viterbi for phase-locked loops.

The book is almost error free. Three minor exceptions are the following. The discussion on page 258 gives the impression that AM modulation is completely general for the phase incoherent channel, which it is not; the claim on page 255 that the simplex signals have been proved optimum is incorrect; finally, the graphs on pages 253 and 265 should have the parameter $\log_2 q$ substituted for q. On the whole, the book provides the student with a background of considerable depth, which no amount of formula manipulation could accomplish. The style, careful attention to clarity, and mathematical level make this book excellent for use as either a course text or for self-teaching.

Steven M. Farber
Information Sciences Lab.
Boeing Scientific Research Lab.
Seattle, Wash. 98124

Perceptrons: An Introduction to Computational Geometry—M. Minsky and S. Papert (Cambridge, Mass: M.I.T. Press, 1969, 258 pp., $12.00 hard cover, $4.95 paper).

This book has been widely hailed as an exciting new chapter in the theory of pattern recognition, but it will be of interest to also view it in another context—as a contribution to the chapter on the complexity of computation by finite networks in the theory of automata.

First, let us recall the idea of a Perceptron. We are to imagine a network made of elements that have a unit delay in their operation and that compute any Boolean function; in other words, given any configuration of 0's and 1's on their inputs, one moment of time later they produce a 0 or 1. (They may be threshold elements, for which each input line has some associated weight W_i, and the overall element has some associated threshold θ. At any moment of time we take the weighted sum $\sum W_i X_i$ of the input values to determine whether the output should be 1 or 0 one moment of time later determined by whether or not that sum exceeds θ.) Later we shall assign other tasks to such networks, but for now let us consider pattern recognition, and consider the input to be a spatially arrayed collection of squares rather than to be a collection of lines. The idea is, then, that we can put a two-dimensional pattern upon this array and it will be quantized so that each square either records a 1, if it is relatively bright, or a 0, if it is relatively dark. Then there is a layer of components that can compute arbitrary Boolean functions, each one connecting to some subset of the squares on the "retina" above. This single layer of associator elements is to be read out by a single threshold element. This structure is the "one-layer" Perceptron studied by Minsky and Papert. (Other authors have studied the effects of interposing more complex networks between the retina and the output element.) In any case, a Perceptron may be used to classify patterns on the retina into those that yield an output 1 and those which yield an output 0.

The question asked by Rosenblatt and answered by many others since[1] is the following. "Given a network, can we 'train' it to recognize a given set of patterns by using feedback, on whether or not the network classifies a pattern correctly, to adjust the 'weights' on various interconnections?" The answers have mostly been of the type, "Well, if a setting exists that will give you your desired classification, I guarantee that my scheme will eventually yield a satisfactory setting of the weights."

I find a certain pleasant irony in the fact that Marvin Minsky and Seymour Papert, long vocal as vigorous opponents of the Perceptron approach to building "intelligent machines," have given the concept a new interest and vigor by giving their essay on computational geometry the title "Perceptrons." In any case, they have now turned the negative comment "Perceptrons are no good for A or B or . . ." into a mathematical examination of the natural class of problems for which Perceptrons are suited. We may say that Minsky and Papert have responded to the questions and answers of earlier workers with the question "OK, you've proved your scheme works when a weighting scheme exists, but when does there exist such a setting of the weights?" In other words, they ask, "Given a pattern recognition problem, how much of the retina must each associator unit 'see' if the network is to do its job?" They analyze this question both for "order-limited Perceptrons," in which the "how much" is the number of retinal units to which an associator unit may be connected, and for "diameter-limited Perceptrons," in which "how much" is specified in terms of the largest diameter of any retinal region to which an associator unit may be connected.

Let us now place this work in the context of complexity of computation, an exciting young branch of automata theory, by first recalling some work of S. Winograd and P. M. Spira, and contrasting it with an example of the group invariance procedure used by Minsky and Papert.

Winograd made a very simple observation that, surprisingly, has most powerful consequences. Suppose that we are using networks made of arbitrary Boolean elements and we look at a box that has two sets of input lines coming in and a set of output lines coming out. We imagine that on the first set of input lines we code the members of some set X_1, on the second set of input lines we code the members of set X_2, and on the output lines we code the members

[1] An excellent review is in Nils Nilsson's monograph, "Learning Machines."

of some set Y. The idea then is that we represent some element x_1 in X_1 on the first set of lines, x_2 in X_2 on the second set, and we hold that configuration of inputs over a period of time. We want the network to be such that after this period the encoding of some function $\varphi(x_1, x_2)$ will appear on the output. Then the question we might ask is, "How long must it take for this to happen?" It is pretty clear that if we allow arbitrarily complicated elements in our network, there will be no problem; but suppose we limit the elements to be such that only have two inputs, and suppose we know that one output line depends on the values of eight input lines (i.e., one of those input lines could not be destroyed without leading to an occasional miscomputation). It is pretty clear that the output line comes from a neuron that can only be affected by at most two inputs and, similarly, each of those two input neurons can only be affected by at most two inputs; so, working back in this way, we see that the computation time required to compute the output cannot be less than three simply because of the problems of fan-in. In fact, if the function is even more complicated, we might well expect more time is required for various things to be moved back and forth.

Winograd was able to show for Abelian groups and Spira was able to show for arbitrary groups that not only did such considerations yield a lower bound on the time required for group multiplication, but, in fact, they could build networks (with elements of the specified complexity, e.g., two input lines per component) that would multiply within one unit of time of that lower bound.

Interestingly, to get this fast computation requires a very redundant encoding of the inputs. In other words, to get the fastest computation in such a network, the input coding is changed in such a way as to spread the information around so that the right information is in the right place at the right time. A simple example of how computation time depends upon encoding is that of multiplying two numbers together. If we multiply numbers in the ordinary decimal notation, all the inputs have to interact. However, if we multiply together numbers that are expressed simply as a string of numbers, with the jth number in the string being the exponent of the jth prime in the prime decomposition of the encoded number, then multiplication becomes much easier. We simply add the exponents. The operation is very fast. However, the encoding is very long, since, if you take a number at random, most primes in a suitable range do not divide it. We thus conclude that there is a big tradeoff between the number of components in each layer and the number of layers or time of computation.

A big interest in the study of complexity of computation is thus to get more insight into the tradeoff between time and space in this way. The work of Winograd and Spira, which we studied, has stressed that, if we bound the number of inputs per component, we can then proceed to discover how many layers of components we require. Minsky and Papert have tackled the complementary problem of asking, "If we fix the number of layers in the network, how complicated must the elements become in order to get a successful computation?" More specifically, rather than regard the sort of algebraic functions (e.g., group multiplication) we have just been talking about, they looked to the problem of classifying patterns presented on the retina of a one-layer Perceptron and asked such questions as "How many inputs are required for the modules above?" Of course, we can always get away with using a single element computing an arbitrary Boolean function and connect it to all the squares. So the question that is really interesting is the following. "Can we get away with a small number of squares connected to each of the input neurons?" Minsky and Papert were able to show, suprisingly enough, that so simple a function as parity—i.e., "give output 1 if the number of squares illuminated is odd, and give output 0 if the number of squares illuminated is even"—requires at least one neuron that is connected to all of the inputs. However, recognition of lines presented on the retina requires associator neurons with only three input lines but very many neurons since we have a line for any three collinear retinal points; whenever two are on, the third must also be on. The output neuron simply detects that no neuron is signalling a violation of this condition.

I do not want to give the proof of the parity result or of other deep results in Minsky and Papert's book, but I do want to prove a very simple result from their book that gives some idea of the flavor of the proof method used. Consider the simple Boolean operation of addition mod 2. If we imagine the square with vertices (0, 0), (0, 1), (1, 1), (1, 0) in the Cartesian plane, with (x_1, x_2) being labelled by $x_1 \oplus x_2$, we have 0's at one diagonally opposite pair of vertices and 1's at the other diagonally opposite pair of vertices. It is clear that there is no way of interposing a straight line such that the 1's lie on one side and the 0's lie on the other side. In other words, it is clear in this case, from visual introspection, that no threshold element exists that can do the job of addition mod 2. However, let us prove it mathematically, because in doing so, we get insight into a general technique that Minsky and Papert use over and over again.

Consider the claim that we wish to prove wrong: there actually exists a threshold element with, say, weights α and β such that $x_1 \oplus x_2 = 1$ if and only if $\alpha x_1 + \beta x_2$ exceeds θ. The crucial point is to notice that the function of mod 2 addition is symmetric and, therefore, it is clear that we must also have $x_1 \oplus x_2 = 1$ if and only if $\beta x_1 + \alpha x_2$ exceeds θ; so, adding together the two terms we have written down, we see that $x_1 \oplus x_2 = 1$ if and only if $[(\alpha + \beta)/2]x_1 + [(\alpha + \beta)/2]x_2$ exceeds θ. Writing $(\alpha + \beta)/2$ as γ we see that, by using the symmetries of mod 2 addition, we reduce three putative parameters, α, β, and θ, to a pair of parameters, γ and θ, such that $x_1 \oplus x_2 = 1$ if and only if $\gamma(x_1 + x_2)$ exceeds θ. So let us set $t = x_1 + x_2$ and look at the polynomial $\gamma t - \theta$. It is a degree 1 polynomial. Let us evaluate it at 0 where we see that we must get $\gamma t - \theta$ less than 0, evaluate it at 1 where we see that $\gamma t - \theta$ is greater than 0, and evaluate it at 2 where we must get a value less than 0. This, we see, is a contradiction. A polynomial of degree 1 cannot change sign from positive to negative more than once. We thus conclude that, in fact, there is no such polynomial, and thus we must conclude that there is no threshold element that will add modulo 2.

We now understand a general method used again and again by Minsky and Papert: start with a pattern classification problem. Observe that certain symmetries leave it invariant. For instance, if it were the parity problem of the simple case of addition mod 2, any permutation of the points of the retina would leave the classification unchanged. We use this to cut down the number of parameters that describe the circuit. We then lump items together to get a polynomial and examine actual patterns to put a lower bound on the degree of the polynomial; we fix things so that this degree bounds the number of inputs to the first layer of the two-layer Perceptron.

"Perceptrons: An essay in computational geometry" is thus a welcome new chapter in the study of pattern recognition. One regrets that this chapter has been presented as somewhat more of a *de novo* contribution than it really is, and that the work was not related to other important studies such as those initiated by Novikoff on the use of integral geometry in pattern recognition. For instance, many contributions in Chapter III are unacknowledged (specifically Kesler's proof of the Perceptron convergence algorithm for more than two categories) although Chapter III is primarily a summary of other work.

But, caviling about history and related work aside, and noting that the book takes longer to make its pertinent points than the reader should be led to think it does, this book is fun to read, full of good ideas about what computer science should be, presents many interesting observations (as: it is pretty impractical to say a Perceptron can classify something if the setting of the weights can be proved to require an accuracy of 1 part in 10^6), and calls important proof techniques to the reader's attention. Anyone who got right through this review and wants more will enjoy finding it in Minsky and Papert's book.

MICHAEL A. ARBIB
Dept. of Elec. Engrg.
Stanford University
Stanford, Calif. 94305