Foreword by Marvin Minsky

Discrete Neural Computation

A Theoretical Foundation

Kai-Yeung Siu
Vwani Roychowdhury
Thomas Kailath

Prentice Hall Information and System Sciences Series
Thomas Kailath, Series Editor

Foreword

This is a wonderful book of stories about the kinds of parallel computers called neural networks. They tell about how much more hardware and time those machines need for doing increasingly larger jobs.

Of course, those kinds of stories wouldn't interest most people. Less, I'd guess, than one in ten thousand. This is a pity because this is also a book about people – because we ourselves are instances of the most advanced parallel machines. Despite that, the theory of machines does not excite the public imagination the ways that sports contests do, or game shows, rock concerts, or the scandals of the private lives of celebrities. No matter. When we're doing mathematics, we don't have to consider the weights of the opinions of majorities.

The book's subject is the computational complexity of problems being solved by parallel machines – and it comes at a timely moment of history. The power of serial computers has grown exponentially for several decades, but in recent years that exponent has started to decrease. Fortunately, at the very same time, the exponent of parallelism has started to grow – while the exponent of memory cost has held its own. This combination of historical trends means that the exponential growth of the past 50 years may well continue long enough to supply us with another trillion fold more of computer power-per-dollar. That log-log curve might even turn up, when we come to the Age of Nanotechnology.

It is now almost exactly 25 years since Seymour Papert and I published *Perceptrons: An Introduction to Computational Geometry*. I see *Discrete Neural Computation: A Theoretical Foundation* as the natural successor to our book. But why did so relatively little happen within that quarter-century interval?

First, we ought to explain why we feel that these machines are so impor-

tant. One reason is the conviction that our very own brains are composed of neural network machines. It would not be very useful to say that a brain *is* a neural network – because that expression is normally used to mean a network of components with a highly uniform architecture. It would be better to describe a brain as a highly evolved assembly composed by interconnecting, in special ways, many different kinds of neural networks. From this point of view, the brain can be seen as like a huge computer with quite a few different and specialized processors, memory systems, and data paths. This means that if we're ever to understand how we work, we'll need good theories both about what each of those networks is able to do, and about how they interact on the larger scale. In the brain, that probably means procedures that are less parallel and more serial.

(Of course, some might argue that this is wrong, that brains are continuous, not discrete. Others might claim that they're infinite. But we need not consider such questions here. When dealing with big mysteries, we should consider the simplest theories first. Philosophical beggars cannot be choosers.)

Neural networks also have practical uses – especially when equipped with procedures for learning from experience. Machines that can learn do not have to be programmed! Of course, we could say the same about any other kind of computer, so why do people emphasize that possibility when discussing neural net machines. Simply, I think, because the knowledge embedded in neural nets is represented in such an exceedingly opaque form. Those collections of numerical coefficients seem, at least on the surface, to be so semantically vacuous that we can scarcely imagine any other practical way to program them!

Now what can such network computers do? In the case of conventional programming for serial computers, we know that everything is possible because, given enough time and memory, those machines can compute any computable function. However, this is not usually the case for neural network machines. Many such systems (and feedforward networks in particular) are not universal computers; what they can potentially compute depends on their specific architectures. Here are some things that we'd like to know for each class C of network machines:

> Competence – What kinds of computations can C-nets be programmed to do?

Learnability – What kinds of computations can C-nets be made to learn?

Learning Time – How long will that learning take to do?

Scalability – How must the size and complexity of C-nets grow in scale, with each type of increase in problem complexity?

Both the present book and the older *Perceptrons* are mainly concerned with Scalability questions. This is because, in the case of networks with a fixed number N of input bits, we can always make a two-layer network machine to recognize any particular subset of such inputs – simply because any such function can be expressed as a disjunction of Boolean conjunctions. Furthermore, if any such recognition is possible at all, it could be accomplished by a trivial kind of "learning" machine that simply tries all possible assignments of coefficients to such a network. Thus Competence and Learnability questions are not interesting unless we ask for quantitative answers about how those answers "scale up" with increasing N. And although Learning-Time questions can be interesting indeed, they make sense only in the case of functions for which the network is Competent.

Earlier I remarked that the opinions of majorities lack weight in mathematical matters. When *Perceptrons* appeared, most theorists acclaimed it. But later there came strong complaints from other neural net researchers, who made remarks like these:

"Those Competence results apply only to 2-layer nets. Multilayer networks can escape those limitations."

Generally, this objection was unjustified, because in *Perceptrons* we almost never considered Competence questions at all, but only Scalability questions about networks under various conditions of bounded fan-in (see Chapter 1). In such cases, the numbers of units and connections still tend to grow exponentially, although having more layers reduces the sizes of exponents. This is often not the case for unbounded fan-ins, and the present book presents many surprising results of this kind.

"Those results may apply to the original Perceptron Learning Algorithm, but things are different now. Today we know how to use Back-Propagation, or Simulated Annealing, or other techniques that transcend those limitations."

This is related to another objection that we frequently heard:

> "They claimed that feedforward networks can't ever learn to compute function X. However, this did not turn out to be true because, in our experiments, the machines learned to do X quite satisfactorily."

The problem word here is "satisfactorily." It has no meaning in the world of pure theory. When we move to the world of practical matters, then we may agree to accept some errors. In pure mathematics we always agree to insist on obeying the rules that we've made – whereas in applied mathematics we agree to consider, instead, systems that work only as well as we need. This in turn raises some of the most difficult – and most important issues in all of science: how to design, and how to test, theoretical models of real-world problems. For example, in practical life, it might be reasonable to ask if a certain kind of network would be good for recognizing faces, or for reading handwriting. There usually is no practical way, then, to agree on a formal definition of such a problem that will always work "perfectly." The trouble, of course, is that those portraits and scrawls are not generated by formal systems with rules that we have access to. For example, how would you start to make a theory of how a machine could distinguish between pictures of dogs and cats? Unless you can set down a practical model that serves to define the boundaries, you cannot begin to prove theorems. The trouble is that terms like "good enough" are never good enough for that.

Once we appreciate this, we may be able to understand what happened to progress in making theories about these machines. It seems to be true that not much happened in the decade immediately after the publication of *Perceptrons*. Thus, the bibliography of the present book has virtually no citations in the 1970s, save for Paul Werbos' prescient thesis. How can we explain those quiet years? Many reporters have repeated the tale that this was because of some gloomy remarks that Papert and I made in our book – which persuaded all those researchers to abandon their work. But this story simply does not ring true. There is no incentive for scientists so strong as the wish to prove that the others are wrong. Another oft repeated tale is that we proved those (still-true) theorems only to get the foundations and sponsors to take those researchers' moneys away. But that compliment is too grand to accept! Mathematicians have no such great powers.

The truth is that it's quite usual for a new mathematical approach to lie fallow for a decade or so. But I think that we did make a different mistake. Our book included, all at once, proofs of almost every easy theorem – and that left almost nothing for the next generation of students to teethe on, so to speak. It was not until the 1980s that the new field of computational complexity had matured enough to supplement, with other techniques, the group-theoretic exploitations of pattern symmetries that we had developed. And it was not until now that, finally, young Kai-Yeung Siu and Vwani Roychowdury, working together with our long-time friend Tom Kailath, managed to bring all those ideas together, to build the almost-unified theory of the complexity of symmetric functions that is displayed in this wonderful book.

There still remains the question of why, in the 1980s, we saw no such theoretical progress coming from the many tens of thousands of enthusiasts who were working with practical neural networks. Scott Fahlman has suggested that although there still was a lack of certain key ideas, this was also due in part to inadequate computer power. To support this, he observes that it was not until Geoffrey Hinton got the backpropagation idea from David Rumelhart and started running examples on his Symbolics machine that it was recognized that the problem of local minima need not be fatal in practice.

It seems to me that another, rather paradoxical factor extended this long delay. It turned out that, once those computations became practical, the feedforward learning machines excelled in many domains in which statistical methods had had little success. This led quickly to useful results, especially in the domain of practical pattern recognition applications. The trouble was that this very success made theory seem less attractive. If your machine failed to work on a certain problem, well, who cares? There are plenty of other fish to fry. So the journals filled up with reports of success – and the failures were rarely mentioned at all. It was exciting to make such discoveries, and rewarding when others found uses for them. Almost no one found time to do theory.

But now we've come to another high peak. What will we next be able to see?

Marvin Minsky
Cambridge, Massachusetts

ELECTRICAL ENGINEERING
Neural Networks

DISCRETE NEURAL COMPUTATION

A Theoretical Foundation

Kai-Yeung Siu
Vwani Roychowdhury
Thomas Kailath

Interest in artificial neural networks has sparked research efforts in many disciplines, including neurobiology, physics, mathematics, computer science, and engineering. To help readers keep pace, this book brings together in one volume the recent developments in discrete neural computation.

Discrete Neural Computation:

- Begins by discussing basic models for discrete neural computation and fundamental concepts in computational complexity
- Establishes efficient designs of threshold circuits for computing various functions
- Develops techniques for analyzing the computational power of neural models

Whether used as a textbook or a reference, **Discrete Neural Computation** presents an integrated approach designed to promote a better understanding of the foundations of neural computation and stimulate further development.

ISBN 0-13-300708-1
90000
9 780133 007084

PRENTICE HALL
Englewood Cliffs, NJ 07632