Dana H. Ballard

An Introduction to Natural Computation

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

An Introduction to Natural Computation

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Complex Adaptive Systems

John H. Holland, Christopher G. Langton, and Stewart W. Wilson, advisors

Adaption in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, John H. Holland

Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, edited by Francisco J. Varela and Paul Bourgine

Genetic Programming: On the Programming of Computers by Means of Natural Selection, John R. Koza

Genetic Programming: The Movie, John R. Koza and James P. Rice

From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior, edited by Jean-Arcady Meyer, Herbert L. Roitblat, and Stewart W. Wilson

Intelligent Behavior in Animals and Robots, David McFarland and Thomas Bösser

Advances in Genetic Programming, edited by Kenneth E. Kinnear, Jr.

Genetic Programming II: Automatic Discovery of Reusable Programs, John R. Koza

Genetic Programming II Video: The Next Generation, John R. Koza

Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds, Mitchel Resnick

From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior, edited by Dave Cliff, Philip Husbands, Jean-Arcady Meyer, and Stewart W. Wilson

Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, edited by Rodney A. Brooks and Pattie Maes

Comparative Approaches to Cognitive Science, edited by Herbert L. Roitblat and Jean-Arcady Meyer

Artificial Life: An Overview, edited by Christopher G. Langton

Evolutionary Programming IV: Proceedings of the Fourth Annual Conference on Evolutionary Programming, edited by John R. McDonnell, Robert G. Reynolds, and David B. Fogel

An Introduction to Genetic Algorithms, Melanie Mitchell

Catching Ourselves in the Act: Situated Activity, Interactive Emergence, and Human Thought, Horst Hendriks-Jansen

Toward a Science of Consciousness: The First Tucson Discussion and Debates, edited by Stuart R. Hameroff, Alfred W. Kaszniak, and Alwyn C. Scott

Genetic Programming: Proceedings of the First Annual Conference, edited by John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo

Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming, edited by Lawrence J. Fogel, Peter J. Angeline, and Thomas Bäck

Elements of Artificial Neural Networks, Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka

From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior, edited by Pattie Maes, Maja J. Mataric, Jean-Arcady Meyer, and Stewart W. Wilson

Advances in Genetic Programming, Volume 2, edited by Peter J. Angeline and Kenneth E. Kinnear, Jr.

Growing Artificial Societies: Social Science from the Bottom Up, Joshua M. Epstein and Robert Axtell

Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems, edited by Christopher G. Langton and Katsunori Shimohara

An Introduction to Natural Computation, Dana H. Ballard

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

An Introduction to Natural Computation

Dana H. Ballard

A Bradford Book
The MIT Press
Cambridge, Massachusetts
London, England

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Third printing, 2000
First MIT Press paperback edition, 1999
© 1997 Massachusetts Institute of Technology
All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

This book was set in Palatino by Omegatype, Inc., and was printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data
Ballard, Dana Harry.
An introduction to natural computation / Dana H. Ballard.
p. cm.—(Complex adaptive systems)
Includes index.
ISBN 0-262-02420-9 (hardcover : alk. paper), 0-262-52258-6 (pb)
1. Brain—Computer simulation. I. Title. II. Series.
QP356.B345 1997
573.8'6'011363—dc21 96-44545
CIP

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Contents

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Figures

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Tables

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Preface

One of the last great challenges is to understand the functioning of the human brain, and it is now clear that the brain is unlikely to be understood without recourse to computational theories. This book attempts to circumscribe the computational material that will form the underpinning of the ultimate set of brain models.

Many excellent texts, from *Introduction to the Theory of Neural Computation* by John Hertz, Anders Krogh, and Richard G. Palmer (Redwood City, CA: Addison-Wesley, 1991) to *Neural Networks for Pattern Recognition* by Christopher M. Bishop (New York: Oxford University Press, 1995), have appeared in recent years. This book differs from both of those in several ways. First, it stresses a broad spectrum of learning models, ranging from neural network learning through reinforcement learning to genetic learning. Second, it attempts to situate the various models in their appropriate neural context. Third, it attempts to make the material accessible to advanced undergraduates as well as beginning graduate students.

Writing about models of the brain before the brain has been fully understood is a delicate matter. One extreme is to make very detailed models of the neural circuitry. The danger that such models need to avoid is losing track of the task that the brain might be trying to solve. The other extreme is to have very abstract models that can represent cognitive constructs that can be readily tested. The danger that these models need to avoid is becoming so abstract that they lose track of all relationships to neurobiology. This book tries to walk a middle ground by choosing a level of discourse that stresses the computational task while at the same time staying near the neurobiology.

The overall theme of the book is that ideas from diverse areas such as neuroscience, information theory, and optimization theory have recently been extended in a way that makes them useful for describing the brain's programs. The many different sources for these ideas are acknowledged at the end of each chapter. In addition I am grateful for permission to use the many figures that illustrate particular points.

Writing any book is a monumental undertaking. I could not have done it without the inspiration of many exceptional doctoral students. The innovative work of Virginia de Sa, Jonas Karlsson, Andrew McCallum, Polly

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Pook, Rajesh Rao, Justinian Rosca, Ramesh Sarukkai, Patrice Simard, and Steve Whitehead lends its sparkle to these pages.

In addition, I have benefited greatly from many discussions with my colleagues at the University of Rochester, especially Christopher Brown and Randal Nelson from the Department of Computer Science and Charlie Duffy, Mary Hayhoe, Peter Lennie, Bill Merigan, Gary Paige, Tania Pasternak, Mark Schieber, and Dave Williams through the Center for Visual Science.

The overall framework for the book has been shaped by many heated discussions by the Woods Hole Workshop on Computational Neuroscience, the brainchild of Terry Sejnowski from the Salk Institute and Joel Davis from the Office of Naval Research, and additionally by experiments made possible through the National Institutes of Health Research Resource at Rochester.

Beside trial runs at Rochester, a draft of the book has been used as a text by Hans Knutsson at Linköping University in Sweden. I am eternally thankful for the work done by Hans and his students in discovering many typographical errors.

A special thanks goes to Peg Meeker who worked with me to refine all aspects of the manuscript and guide it through its LaTeX birth. We both rave about Textures, an excellent piece of software for the Macintosh. We could not have succeeded, moreover, without the software expertise of Liudrikas Bukys.

I have been generously supported by the National Science Foundation through Howard Moraff and by the National Institutes of Health through Richard DuBois.

Last, I would like to thank Janie for our long walk together. I love you.

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

An Introduction to Natural Computation

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

1 Natural Computation

1.1 INTRODUCTION

One of the most exciting challenges of our time is to understand animal intelligence. Such a quest necessarily has as its focus a model of animal brains and most importantly a model of the operation of the human brain.

> The central tenet of modern neural science is that all behavior is a reflection of brain function. According to this view . . . what we commonly call mind is a range of functions carried out by the brain. The action of the brain underlies not only relatively simple motor behaviors such as walking, breathing and smiling, but also elaborate affective and cognitive behaviors such as feeling, learning, thinking and composing a symphony.[1]

Enormous progress has been made toward understanding the brain at many different levels, from the molecular to the biophysical. We now are zeroing in on detailed models of neurotransmitter chemistry and genetic structure. As we begin to understand the complexity of these underlying processes, a growing consensus is that *computational models*[2] will play a vital role in a complete picture of brain function, particularly in modeling more macroscopic structures that may be directly related to our conscious perceptions.

Computational models seek to characterize how the brain acquires and uses information in behavior. Interest in computational models, particularly in computational models of learning, has virtually exploded in the past decade for a number of reasons. One is the development of an array of new computational methods. Another is the continuing improvement in the speed of computers, which has made the simulation of very large systems possible. A third reason is the cross-pollination of computational models with experimental methods; brain-imaging methods are developing to the point where computational models can be directly tested. The purpose of this book is to serve as an introduction to the computational models that are spurring these new developments.

That the brain could be described by computation is itself controversial. The issue is that computation is a weak model, and, as we will see in a moment, most of the key theoretical results from computer science are about limitations. The key question, however, is, Is computation sufficient to

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

model the brain? One view is negative, and this is articulated in Roger Penrose's critiques.[3] Penrose thinks that the extra structure of quantum physics may have additional power. This text, on the other hand, articulates a more mainstream and positive view: Computation is a weak model, but emergent evidence suggests that it may be good enough. Taking just two examples:

- A driving algorithm learns correspondences between road images and steering commands.[4] On its first tests, it drives Carnegie Mellon University's automated van ten times faster than had previously been possible.
- A game-playing algorithm learns to play world-class backgammon, having tied the world champion.[5] A key element is that it can play itself and constantly improve.

While these algorithms cannot violate the laws of computation, they do make use of a number of computational insights that break with the tenets of classical computing. These insights stem directly from the use of computational models to understand brain function, and they form the focus of this book. To highlight them, we use the title *An Introduction to Natural Computation,* borrowing a phrase coined by Whitman Richards.[6] Richards's goal was to describe the necessary inferences a biological system must make given that its computational abilities are limited. There could be no better introduction to the scope of this book:

> Biological systems have available through their senses only very limited information about the external world. Yet these systems make strong assertions about the actual state of the world outside themselves. These assertions are of necessity incomplete. Clearly, a replica of an object and its qualities cannot be embodied within the brain. How can an incomplete description, encoded within neural states, be sufficient to direct the survival and successful adaptive behavior of a living system?[7]

1.2 THE BRAIN

The brain is an exquisitely beautiful structure that has evolved over millennia to perform very specific functions related to animal survival and procreation. It is a very complex structure that can be understood only by studying its features at different spatial scales.[8]

1.2.1 Subsystems

At the largest scale, the brain has specialized mechanisms for input, output, short-term memory, long-term memory, and arousal. The functions of the major subsystems are far from completely understood, but Table 1.1 will serve to orient you.

When it comes to computation, the most important parts of the brain are the cerebral hemispheres and their immediately associated structures. Each of these has a special function as shown in Figure 1.1.[9]

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Table 1.1 The organization of the brain into a layered anatomical structure with increasing refinement of function culminating with the site of complex programs and memory in the cerebral hemispheres.

Component	Function
Cerebral hemispheres	The cortex, basal ganglia, hippocampus, and amygdaloid nucleus form the site of the brain's complex programs and memory
Diencephalon	Thalamus: cortical I/O; hypothalamus: regulation of autonomic, endocrine, and visceral function
Midbrain	Eye movement control; visual and auditory reflexes
Medulla oblongata	Peripheral circuitry to control autonomic functions, e.g., digestion, breathing, heart rate
Pons	Interface between cerebellum and cerebral hemispheres
Spinal cord	Peripheral circuitry to control movement of limbs and trunk; receives and processes sensory information

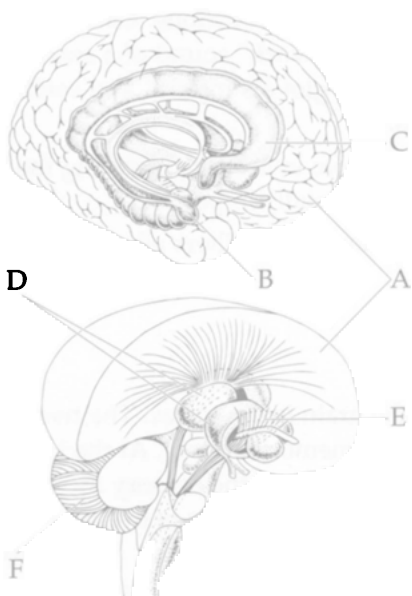


Figure 1.1 The basic function units of the human brain: (*A*) cortex: associated with long-term memory; (*B*) amygdala: associated with arousal or characterizing the importance of the current processing; (*C*) hippocampus: associated with recording working memory, that is, the variables used in the current programs; (*D*) thalamus: associated with input and output modulation; (*E*) basal ganglia: associated with the establishment of sequential programs; contains a prominent chemical reward system; (*F*) cerebellum: associated with motor "memory" or learned sensorimotor subroutines. (From Pansky and Allen, 1980; reproduced from *Review of Neuroscience*, © 1980, Macmillan, with permission of the McGraw-Hill Companies.)

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

• The *cortex* is the main site of the brain's permanent memory. It has the structure of a six-layered sheet of neurons and is specialized into different regions. The cortex is the most studied part of the brain, and the most studied region of cortex is that at the back of the brain, which is responsible for vision.

• The *basal ganglia* is a region in the center of the brain that plays a major role in sequential actions that are central to complex behaviors. It has elaborate chemical reward systems for rating the value of different action sequences. These are essential as a fundamental problem is estimating the value of current behaviors that are done for future rewards.

• The *hippocampus* plays a central role in the permanent recording of momentary experiences. Of the deluge of ongoing experiences in the brain, what is worth remembering? The hippocampus has mechanisms for registering current experiences until they can be stored more permanently. For people who have injured their hippocampus, time stops at the point of injury; they can participate in conversations, behaviors, and the like but do not remember them.

• The *cerebellum* plays the major role in the memory for complicated sensorimotor experiences associated with actions. Catching a baseball in a glove requires associating the "thwack" sound of a successful catch with motor actions that control the glove's inertia. The cerebellum handles these complex associations.

• The *amygdala* plays a major role in arousal, orienting the brain to place emphasis on events that are especially important.

• The *thalamus* is a major gateway that filters all sensorimotor input and output to the cortex.

1.2.2 Maps

The most studied part of the brain is the cortex. The two hemispheres of the cortex form a hierarchical memory system. At the lower levels of the memory are simple representations of the sensory input and motor output. Higher levels represent abstract features of the same that are used for problem-solving and control functions. In vision, for example, the lowest part of the cortical hierarchy represents "edges," local sites of abrupt photometric change in the image encoded by the retinas. The highest parts of that same hierarchy represent features of faces, such as emotions and identity. The cortices form a two-dimensional layered sheet of nerve cells that can be grouped together in *maps* wherein cells have common functions. Figure 1.2 shows the basic layout of a monkey cortex that has been flattened to make the map structure more apparent.[10]

1.2.3 Neurons

The major components and cortical maps are architectural features that show up at large and intermediate scales, respectively. A much more stun-

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

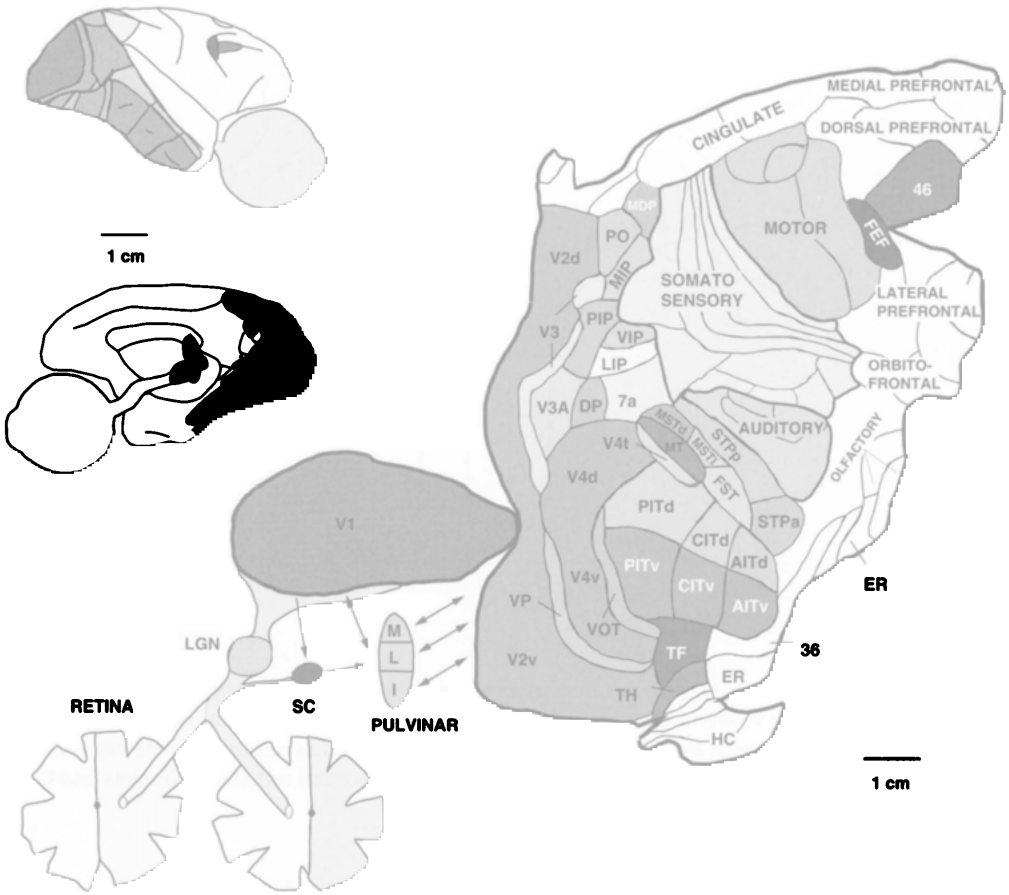


Figure 1.2 The brain's main memory system is the cortex, a two-dimensional multilayered sheet of cells that appears in folds directly underneath the skull. These are specialized into different functions that are laid out in two-dimensional "maps" or collections of cells that can be associated with distinct functions. Although such functional interpretations are constantly being refined and revised, the major areas are as shown. (From Van Essen, Anderson, and Felleman, 1992.)

ning feature shows up at a small scale, and this is the brain's capacity for huge amounts of parallel computation by means of nerve cells, or *neurons*. Neurons come in a small variety of basic shapes, but the main type used for communicating over distances is the pyramidal cell, whose principal features are shown in Figure 1.3.[11] Each of the approximately 10^{11} cortical neurons connects to 10^4 others. This connectivity is nothing short of amazing owing to the special "tree" shape of a neuron. Consider that it is not unusual for a neuron, whose cell body is about 10 microns, or 10^{-5} meters, to connect to neurons one centimeter distant. If the cell body were the size of a marble, its axon would stretch 30 feet. This axon branches more than 10,000 times in the course of making connections,[12] and each one of these branches is capable of sending the cell's signals, which are in the form of voltage spikes. At the end of each axonal branch, the signal is transmitted

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

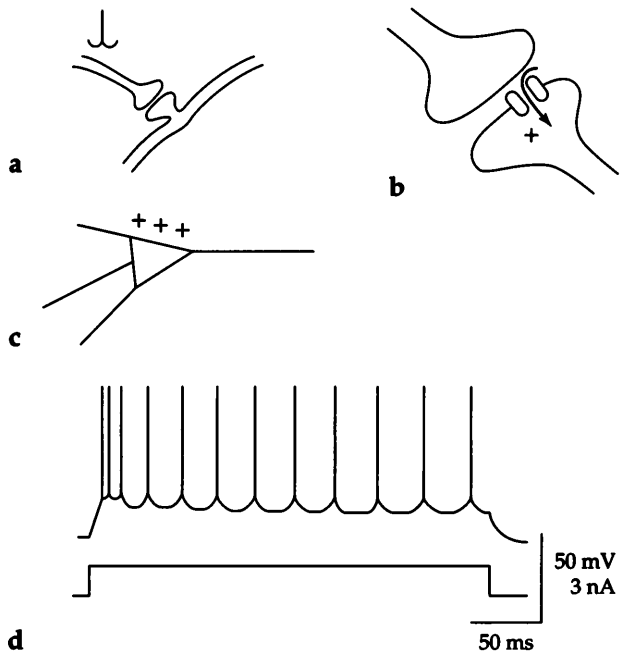


Figure 1.3 The basic features of a neuron showing how a neuron communicates: (*a*) an incoming voltage spike arrives at a synapse; (*b*) this causes ions to flow across the synaptic junction, charging the cell body; (*c*) when sufficient charge has accumulated, the cell fires its own voltage spike. (*d*) Typical spike sequences from a cortical cell. (From Connors and Gutnick, 1990.)

to another cell by *neurotransmitters*, chemical messengers that effect a charge transfer. The site of transfer is called a *synapse*. Synapses on the receiving cell are located on *dendrites*, extensions of the cell near the cell body. In the learning algorithms that are the focus of natural computation, synapses are the central feature, as modulating their efficacy is thought to be the principal way brain programs are constructed.

Think of each neuron as a processor that can compute in parallel with all the other neurons. If the computation could be optimally distributed over these processors, then potential speedups on the order of the number of neurons could be achieved. This speedup is vital because the brain is under even more severe constraints than silicon computers, since its basic computing units—neurons—are more than one million times slower than silicon circuitry.

1.3 COMPUTATIONAL THEORY

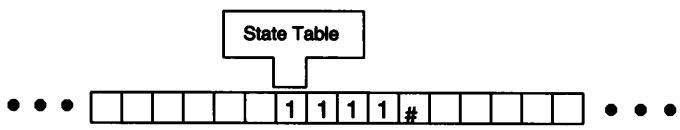
If the brain is performing computation, it should obey the laws of computational theory. These results come from two areas, *computability* and *complexity*, and can be paraphrased respectively as follows: (1) You cannot compute nearly all the things you want to compute. (2) The things you can compute are too expensive to compute.

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Computability seeks to characterize the entities that can be computed. Computation can be modeled on a universal machine, called a Turing machine. Such a machine has a very simple specification, as shown in Figure 1.4. The machine works by being in a "state" and reading a symbol from linear tape. For each combination of state and tape symbol, the machine has an associated instruction that specifies a triple consisting of the new state, a symbol to write on the tape, and a direction to move. Possible motions are one tape symbol to the left or right. Although the Turing machine operation appears simple, it is sufficiently powerful that if a problem can be solved by any computer it can be solved by a Turing machine.

The things that Turing machines compute are called *computable functions*, and from examining the power of Turing machines, you can learn

a)



b)

Situation		Action		
Read	State	Move	New State	Write
0	0	R	0	0
1	0	R	1	0
1	1	R	1	0
#	1	R	Halt	0

c)

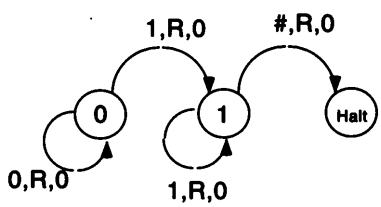


Figure 1.4 (*a*) A Turing machine consists of a linear tape containing a discrete set of symbols. The machine reads the symbol immediately below the tape reader head and, depending on its state, writes a symbol and moves one tape space to the left or right. A Turing machine is completely characterized by its state transition diagram. This example shows a very simple program for erasing a block of contiguous ones. (*b*) The program can be represented by a table that shows what to do for each state and input. (*c*) Equivalently, a Turing machine program can be described by a state transition diagram in which the nodes of a graph are states and arcs are labeled by the symbol read, the direction of motion, and the symbol written. Despite the extreme modesty of its structure, a Turing machine is sufficiently powerful to be able to emulate all the operations of any other computer, albeit much less efficiently.

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

properties of these functions. The most essential property stems from the fact that Turing machines can be enumerated in correspondence with the integers. Thus the only computable functions are those that can also be put in correspondence with the integers, a countable infinity. This is indeed a limitation since the set of *all* functions can be put in correspondence with the real numbers, an uncountable infinity. Thus there is an uncountably infinite number of uncomputable functions. This is not hard to show formally. Putting the Turing machines in correspondence with the integers only requires that you do not use up the integers too fast, and is left as an exercise. Showing that the integers and reals cannot be put into correspondence can be done by assuming the contrary and then showing that such an assumption leads to a contradiction. First you hypothesize that it can be done for the real numbers between zero and one, and then you find other real numbers that have been left off the list, as shown in Figure 1.5. Therefore, it cannot be done. Hence Turing machines must operate within this limitation.

Complexity seeks to characterize the *cost* of computer algorithms. The way this cost is measured is as a function of the size of the input. For example, sorting a list of n numbers takes time proportional to $n \log n$. Multiplying two square matrices together, each of size $n \times n$, takes time proportional to $n^{\log 7}$. These two examples are regarded as "easy" because their run time is bounded by a polynomial function of their input; that is, you can find a constant k such that the run time $T(n) \leq k f(n)$ where $f(n)$ is a polynomial function of n.

Unfortunately for most of the interesting computable problems that have a "best" answer, complexity theory tells us that the cost of computing

Integers	Real Numbers between 0 and 1
1	.0XXXXXXXXXX...
2	.X3XXXXXXXXX...
3	.XX8XXXXXXXX...
4	.XXX2XXXXXXX...
5	.XXXX1XXXXXX...
6	.XXXXX3XXXXX...
.	.
.	.
.	.

Figure 1.5 A hypothetical experiment that will not work. Suppose that the integers can be put in one-to-one correspondence with the real numbers between zero and one. In this example Xs are used to stand for arbitrary digits. Now one can make a new real number that is not on the list by making it differ from the first number in the first significant digit, the second number in the second significant digit, and so on. Therefore, it cannot be on the list.

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Table 1.2 Time to solve a problem on a computer that takes 10^{-7} seconds per unit input as a function of the size of the input and the complexity of the algorithm. Times are in seconds except where noted. For *polynomial* complexity functions such as n and n^3, the times remain reasonable. *Exponential* complexity functions such as 2^n and 3^n can only be solved for very modest n.

Algorithmic Complexity	Input Size, n			
	10	30	50	70
n	10^{-6}	3×10^{-6}	5×10^{-6}	7×10^{-6}
n^3	10^{-4}	0.27×10^{-2}	0.12×10^{-1}	0.34×10^{-1}
2^n	0.2×10^{-4}	61 hr	29×10^{13} yr	14×10^{32} yr
3^n	0.18×10^{-1}	15×10^{9} yr	47×10^{29} yr	49×10^{60} yr

this answer is prohibitively expensive, since the run time scales as an exponential function of the size of the input, that is,

$$T(n) = kC^n$$

It does not matter what the constants C and k are, so suppose $k = 1$ and $C = 10$. Then if a problem of size 100 can be solved in one second, a problem of size 10,000 would take 10^{100} seconds, or much longer than the age of the universe. Table 1.2 shows additional calculations, revealing the sharp boundary that separates problems that have polynomial run-time functions and problems that have exponential run-time functions. Most interesting problems have exponential worst-case functions, so that the solutions to their worst-case exemplars, for all but the smallest problem sizes, are too expensive to compute.

1.4 ELEMENTS OF NATURAL COMPUTATION

The result of computability theory is fundamental. But the result of complexity theory turns on a series of assumptions that range from the architecture of the machine to the size of the problems to be solved to the criterion to be used in evaluating solutions. Natural computation breaks with all these assumptions and replaces them with a new modeling perspective. The elements of this perspective are as follows:

• *Minimum description length.* The only answers that are practical to compute are those that retreat from the best answer in some sense. Answers can be just good, approximately correct, or correct to a certain probability. A universal metric for all these approximations is the minimum-description-length principle, which measures the cost of encoding regularity in data.

• *Learning.* Biological systems can amortize the cost of algorithms over their lifetime by learning from examples. Such learning can be seen as the "online" detection of regularity. The crucial component of this online

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

performance is the ability to *predict* future events, an ability that has enormous survival value.

• *Specialized architectures.* The massively parallel organization of the brain's neurons can compensate dramatically for their millisecond speeds. Particularly if the input is bounded at some fixed size, as it is with a retina or cochlea, then it can be very cost effective to design special-purpose architectures. In addition, to manage complexity the brain has evolved many hierarchical structures.

These broad elements have led to the development of a wide range of computational models that both are effective and provide many new insights into the possible mechanisms of brain computation. These models lie properly at the boundary of mathematics, computer science, psychology, and neuroscience. Let us elaborate.

1.4.1 Minimum Description Length

Although the results from complexity theory on computational cost are daunting, the key caveat is that they are for the "worst-case" assumptions. Worst-case assumptions mean that the very best result is to be guaranteed, regardless of the difficulty of any particular data set. Figure 1.6 shows that instances of a problem can vary hugely in difficulty. In the traveling salesman problem the challenge is to find the shortest tour of n cities, returning to the point of origin. As the figure shows, some problems are easy enough that very good solutions can be found readily, whereas others may be so difficult that they require testing all possible routes.

Retreating from this standard in any of several directions makes computation practical. For example, the solution may be close to the optimum in cost, but not the best. In the traveling salesman problem, a tour with a length within 10% of the shortest length may do. Another way of reducing cost is to have a solution that has a high probability of being the best, but may not actually be the best. Yet another way of approximating is to have

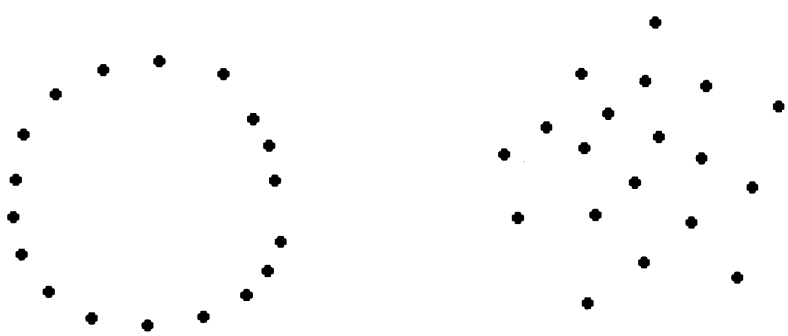


Figure 1.6 Easy (*left*) and difficult (*right*) traveling salesman problems. Finding the minimum tour of the cities can require examining all possible tours in the general case, but getting within a percentage of the best tour is inexpensive in both cases.

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

a solution that is almost correct. In a biological system that takes time to implement solutions, a partial solution may be satisfactory, as the rest can be filled in later on.

Approximate solutions are almost forced on biological systems because they have to act continuously in the world, and new solutions have to be found by experimentation, either directly or by mental simulation. Since the experimental trials cannot be guaranteed to be the best, the system will necessarily be suboptimal. In the words of Stephen Jay Gould: "In order to be adaptable, an organism must be suboptimal."[13]

In thinking about approximate algorithms it is important to consider their physical embodiment. Something as ordinary as a table might be thought of as running an algorithm that adjusts its atoms continually, governed by an energy function. Whatever its variables are, just denote them collectively by x. Then you can think of the table as solving the problem of adjusting its atoms so as to minimize energy, that is,

$$\min_{x} E(x)$$

Is this computation? It turns out that it is tricky to say. Some contend that computation is not a *natural kind*,[14] that is, whether or not some device is doing computation depends on the goals of an external observer. This may turn out not to be the case. Extraordinary progress is being made by casting computation in terms of an optimization problem in which some function of the variables of the system is to be minimized. Exploring possible functions is in large part what this book is about.

One of the most promising cost functions views the brain's algorithms as compact codes. This idea stems from the work of Kolmogorov[15] in the 1930s. Consider that a system can always be defined in terms of the sets of input/output data it produces. If these sets are random, then it turns out that this is the best one can do. That is, there is no smaller description of the data than the data itself. However, physical systems incorporate many regularities, and, as a consequence, much more economical descriptions are possible. In particular, small descriptions are created by the brain's encoding mechanisms.

When counting the size of encoded data to see that you really made a net improvement, you have to count the size of the encoder as well. For animals, the encoders are in the form of their behavioral programs. Thus a new version of Occam's razor is that the complexity of a theory (read: corpus of behavioral programs) can be measured in terms of the minimum-description-length principle, which counts succinctness in terms of the compactness of the theory in both the data it accounts for and the complexity of the theory itself. The only step left is to specify a currency for this accounting. The standard units for encoding are *bits* from information theory. Informally, you can think of a bit as the amount of information you obtain from learning the result of an experiment with two equally probable outcomes. In these units the MDL principle is easy to express (see box).

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

> The complexity of a theory is measured in terms of the number of bits to encode the theory + the number of bits to encode the data with the help of the theory.[16]

Let us illustrate this principle with two examples.

Example 1: A Program That Prints 10,000 Ones Suppose the data set is the string of 10,000 ones,[17] that is,

1111···1

These can be compressed by using the equivalent program

```
for i=1 to 10,000 do
   print(1)
```

In this example, the "theory" is a simple program that can be described in just a few hundred bits. Once this program has been created, the "data" are compressed into the null set. In more complicated cases, simpler theories can usually be obtained by not trying to predict all the data exactly but leaving an uncompressed part as a residual, as in the next example.

Example 2: A Neuron's Receptive Field The main form of programming is in terms of the modification of a synapse.[18] To see how this works, consider the example of encoding a small patch of an image by a set of model neurons. Although real synapses are complicated, much insight can be had with a simple model in which their effects are just a multiplication of the synaptic strength with the input. That is, if x is the value of the incoming neuron's input, and the weight w models a synapse on the target cell, the effect of the input is wx. In this simple model, the values of a model neuron's synapses play a major role in specifying how it will respond to its input. Its collection of synapses constitutes a code that makes it especially sensitive for inputs that have the same pattern of activity as the strength of the weights. Figure 1.7 shows how this can be done. The responses of a set of neurons constitute a code that takes into account regularities in the visual world to compress the image samples into a much smaller format.

The minimum-description-length principle is a breakthrough because now one can think of a program as a form of code and the job of the brain's self-organization (algorithms) as producing compact codes. You can view behavioral programs as operating with *compressed descriptions* of data. Theories can be rated according to how much they compress the data.

The value of compact codes is evident in the rapid expansion of brain size in hominid evolution, which presumably pays off in increased fitness. To survive, the brain needs to simultaneously respond quickly to situations and have a large repertoire of responses. Both these avenues have to deal with biological constraints. The speed of responses is limited by the

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

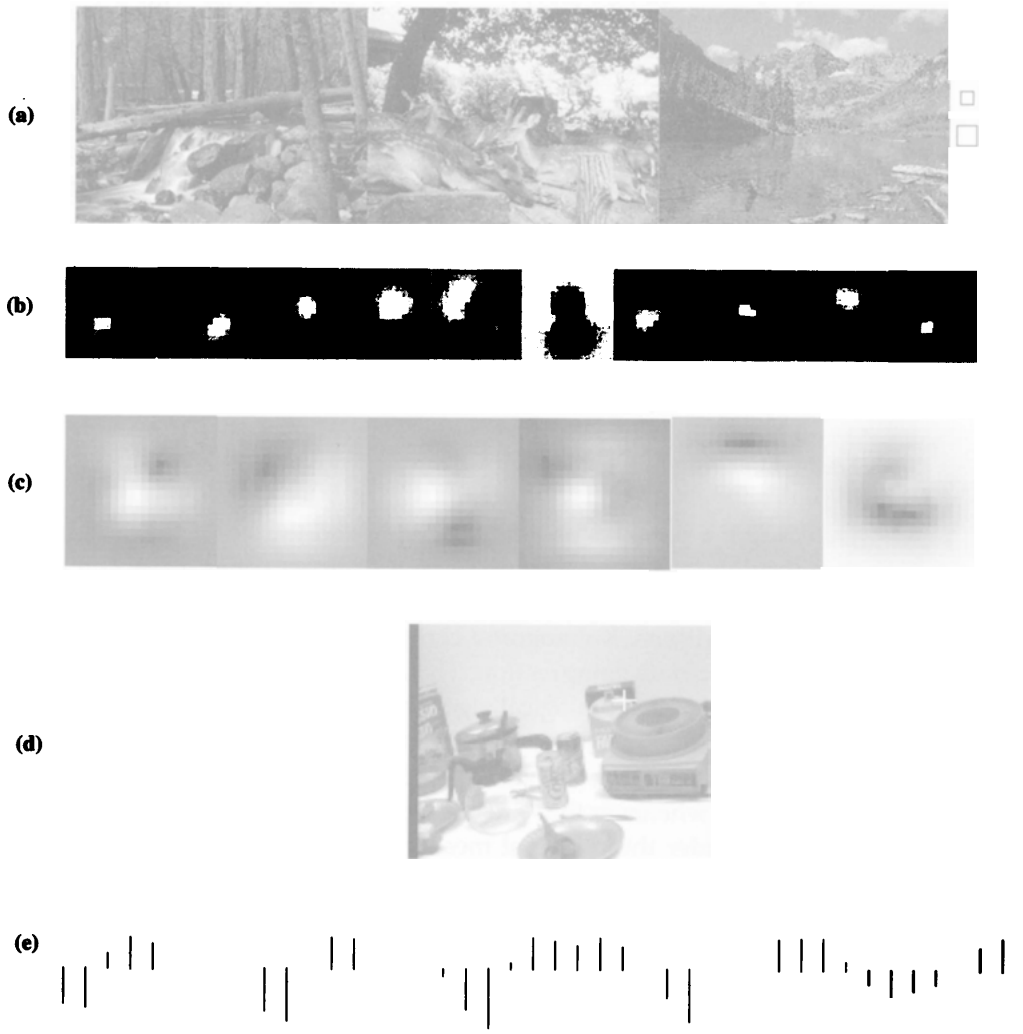


Figure 1.7 How a neuron can represent a code for images. (*a*) 16 × 16 and 20 × 20 sample patches from natural images whose relative size is shown as small boxes on the right are input to 16 neurons' synapses or *receptive fields*. (*b* and *c*) The synapses of 16 model neurons shown with the two receptive field sizes. The weights for the 16 × 16 and 20 × 20 synapses can be both positive and negative numbers. Positive numbers are encoded as lighter squares, and negative numbers are encoded as darker squares. Each of the 16 "neurons" will respond according to how well a particular input matches its receptive field. A collection of such responses constitutes a code that describes the image patch more succinctly than the original samples. (*d*) The portion of a new image centered on the cross to be succinctly encoded as the collective response of 45 such neurons. (*e*) The bars show the firing rate of individual neurons from a horizontal baseline rate. (Parts *d* and *e* are reprinted from *Artificial Intelligence*, 78[12], Rajesh P. N. Rao and Dana Ballard, "An Active Vision Architecture Based on Iconic Representation, p. 473, © 1995 with kind permission of Elsevier Science–NL, Sara Burgerhartstraat 25, 1055 KV Amsterdam, The Netherlands.)

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

speed of neurons, which cannot signal faster than one millisecond. The size of the repertoire is limited by the number of neurons. (In humans, brain size has exceeded what is practical to push through the birth canal, and flexible skull plates permit postnatal growth of the brain size of up to a factor of four.[19]) Thus the simplest way to improve things is to discover better encodings (algorithms), and the way to measure these is in terms of their information in bits.

It can be extremely disconcerting to think of human attributes such as goals and emotions as being measured and driven by encodings which are in turn measured in bits, but this stance is very helpful if we are to understand the brain. It turns out that goals and emotions can be seen as having essential algorithmic roles in the execution of behaviors.

1.4.2 Learning

Thinking of a program as a form of code and the job of the brain's self-organization (algorithms) as producing compact codes gives you a universal perspective on the brain's algorithms. These are characterized as *learning* algorithms. Kolmogorov complexity provides algorithms with a common task: data compression. The key additional constraint captured by "learning" is that biological algorithms must work *online*. That is, they function continually in the embodiment of the organism. In computer science parlance, they are "anytime" algorithms in that the behaviors must be produced whenever they are needed.

Now consider the principal mechanism of learning. Each of approximately 10^{10} neurons connects to an average of 10^4 other neurons by means of synapses.[20] The principal way of programming the brain is to change the strength of these synapses, as each of the brain's 10^{14} synapses is modifiable. This modifiability gives the brain the capability of continually reprogramming itself to meet current requirements. Learning algorithms exploit this underlying plasticity to amortize the cost of their development. In some cases, this amortization extends far beyond the lifetime of a single individual. This book focuses on three kinds of learning—developmental learning, behavioral learning, and evolutionary learning—all of which affect the architecture of the brain. What distinguishes them is that they operate on very different timescales.

Developmental learning occurs just before and soon after birth and requires on the order of weeks to months. In visual development, for example, stereo vision appears at about five weeks after birth. Most developmental learning occurs in the first year, but complex changes occur on a schedule that continues up to puberty. In terms of brain circuitry, this is the time during which the basic connections are made. The amount of plasticity is enormous. It is estimated that, in the course of searching for the right neural connections, ten times the number of cells that survive puberty are tried out and discarded. The goal of this learning is to process basic

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

stimuli from the environment in order to react quickly. The way this can be done is to associate complex invariants from the environment with actions in a look-up table format. For example, "tiger" has a myriad of sensory instantiations, but the goal is to recognize the essential invariants and associate them with "run!" in the look-up table.

Behavioral learning occurs at the task level. The agent learns to string together primitive physical actions that are encoded in look-up table format to accomplish goals. The key feature of this kind of learning is that the rewards are delayed. Thus the agent has to solve the credit assignment problem: learning which of many different causes at a particular time lead to sources of reward at a later time. Developmental and behavioral learning overlap in time, but developmental learning must occur first, as a set of primitive functions must be in place to get behavioral learning started. Behavioral learning continues throughout one's lifetime.

Evolutionary learning occurs at much longer timescales. By experimenting with the structure of genes, animals can test modifications in architecture. We know that the first bipedal creatures like us appeared about 3 million years ago but that *Homo sapiens* may have appeared as recently as about 200,000 years ago.

Somewhat surprisingly, the resources available to do computation in each of the learning categories are similar. The brain is programmed from experience. At different timescales, experimenters have observed the time needed to acquire certain facilities, so we can estimate an upper bound for the training time for a given level. In general, the bound on the training time T_{train} is given by the simple formula

$$T_{train} = T_p \times N_p$$

where T_p is the time to process stimuli from the environment and N_p is the number of such events required to learn. This formula can be used with the observations of T_{train} from the previous paragraph to estimate the number of presentations for each of the learning types. The results are shown in Table 1.3. What is interesting is that the estimates for N_p are all within a couple of orders of magnitude of each other, indicating that the power of the algorithms is about the same.

Table 1.3 Learning times for human brain processes. Although the timescales of operation are very different, the power of the underlying learning algorithms may be comparable. The power of evolutionary learning is underestimated here, as the search is conducted with a population rather than an individual.

Learning Venue	Exposure Time, T_p	Total Time, T_{train}	Learning Epochs, N_p
Development	10^{-1}	10^7	10^8
Behavior	10^1	10^9	10^8
Evolution	10^{10}	10^{16}	10^6

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

1.4.3 Architectures

One way of reducing computational cost is to design the computer in such a way as to be specialized for the computations that must be performed. This might seem a violation of the main result of complexity theory: that the cost of an algorithm should be measured only as a function of the size of the input and nothing else. The key difference is that biological systems have finite input sizes. Thus the complexity arguments, which hold in the limit of arbitrarily large inputs, do not necessarily apply. When the input is of finite size, it pays to optimize the machine.[21]

Communicating in physical systems over long distances has costs in time and space.[22] There is great incentive to organize computation so that most of it is done by communicating *locally*, but in any complex system some long-distance communication will be needed. The only way around this problem is to organize such systems hierarchically. This approach allows one to tailor effects so that they can be reliably accessed at the correct temporal and spatial scale. (One can have systems for which local effects can be of great consequence at long scales. Such systems are termed *chaotic*.[23] But these systems cannot be easily utilized in computation.) The bottom line is that for any physical system to be manageable, it must be organized hierarchically.

Constraints of Time and Space Hierarchical systems have a fundamental constraint that stems from the organization of space itself. Whenever a system is constructed of units that are composed of simpler primitives, the more abstract primitives are necessarily larger and slower. The effects of spatial scale are shown in Figure 1.8. Suppose the 0th level of a system is constructed of components of size C that may be thought of as primitives. Then the next level in the system that uses K of these will take up space KC. Moving up the hierarchy, the nth level will cost K^nC.

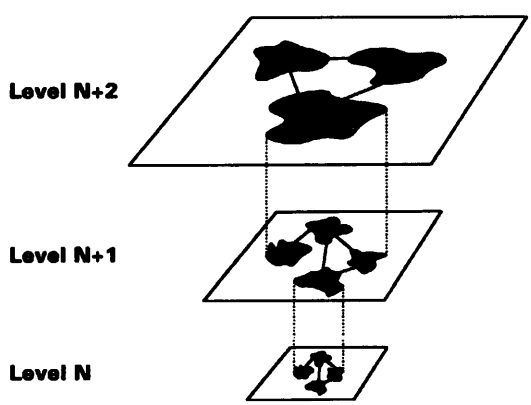


Figure 1.8 Spatial scaling in hierarchical systems. Composite units each take the sum of the space of their components. Thus more abstract circuits necessarily take up more space than their components. (From Newell, 1990.)

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Time scales in a similar way, as shown in Figure 1.9, but the reason time may not scale geometrically is that temporal costs can be ameliorated with special algorithms. These usually use special connectivity so as to push temporal costs into the spatial domain. Thus instead of a cost of tM for level n, the actual cost will be T^nM, where $T = f(t)$ is the temporal scaling owing to the special algorithm.

We have just seen that there are fundamental spatiotemporal constraints on hierarchical systems. In a hierarchical system, the more abstract components run slower and take up more space at geometric rates.

Cognitive Hierarchies The temporal constraints on levels in a hierarchical system can be interpreted in terms of biology.[24] The most fundamental constraint for the human brain is the communication system between neurons. Neurons communicate by sending electrical spikes that take about one millisecond to generate. As a result, the circuitry that uses these spikes for computation has to run slower than this rate. Let us assume that about ten operations are composed at each level. Then local cortical circuitry will require 10 ms. These operations are in turn composed for the fastest "deliberate act." A primitive deliberate act is then 100 ms. Measurements of perception show that such acts take about 50 ms, so the 100-ms estimate is in the ballpark. The next level is the physical act. A primitive physical act might be an eye movement, a hand movement, or a short sentence. Composing these results is a primitive task. A good example is a chess move. Speed chess is played at about 10 seconds per move.[25] Table 1.4 shows these relations.

To summarize the discussion of hierarchies, there are two important points. The first is that the abstract analysis of hierarchical systems finds a reality in the construction of the brain. The evidence is overwhelming that the brain is organized hierarchically. The second point is that the constraints that the individual levels impose on models of computation are quite severe. For example, an eye movement takes about 250 ms (a little shorter than Newell's estimate). To make that movement, the neural machinery must choose the eye-movement target, locate that target in space, and command the eye muscles, all in the time it takes a neuron to send about 100 spikes. This in turn means that many of the operations must be

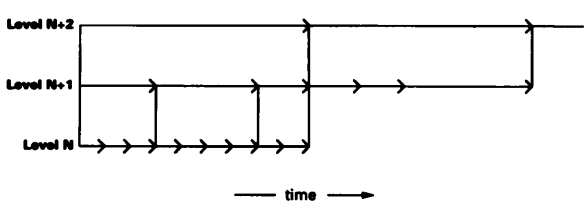


Figure 1.9 Temporal scaling in hierarchical systems. Composite units each take the sum of the time of their components. Thus more abstract operations are necessarily slower than their component operations. (From Newell, 1990.)

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Table 1.4 The organization of human computation into temporal bands (after Newell, 1990). Though the cost of developing algorithms may be amortized over a much longer period, ranging from years for developmental learning to centuries for genetic learning, the timescale for their execution is tightly constrained to lie within a range of 100 ms to 10 seconds.

Temporal Scale	Primitive	Example
10 sec	Complex task	Moving in speed chess
2 sec	Simple task	Saying a sentence
300 ms	Physical act	Moving the eyes
50 ms	Neural act	Noticing a stimulus
10 ms	Neural circuit	
1 ms	Neuron spike	

precomputed, as there is not enough time to synthesize elaborate sequences of neural "instructions."

1.5 OVERVIEW

The overall perspective of this book is that learning algorithms develop behavioral programs. Such algorithms can be seen as being driven by the minimum-description-length principle. The brain works to improve its owner's chances of survival by working simultaneously to increase the speed of execution of its programs and the size of its behavioral repertoire.

This book brings together three main types of learning algorithms: neural networks, reinforcement learning, and genetic algorithms. These algorithms operate at very different timescales, but they share a common theme in that they are all optimization algorithms that try to find good solutions to problems using examples as input. Although these algorithms can be useful in many situations, much leverage is gained by studying them as models of brain computation. Thus developmental learning is developed with the objective of encoding quick reactions to stimuli. Reinforcement learning is developed with respect to the brain's problem of using secondary rewards. Genetic algorithms are developed in the context of searching the space of brain architectures. Table 1.5 shows the correspondence used in this text.

Table 1.5 The correspondence between learning algorithms and learning systems in the brain.

Biological Timescale	Computational Model	Function
Development	Neural networks	Memories
Behavior	Reinforcement learning	Programs
Evolution	Genetic algorithms	Architectures

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

The structure of the book has four parts: a set of core concepts (Part I), followed by the study of the learning algorithms in three settings of increasing abstraction: memories (Part II), programs (Part III), and architectures (Part IV).

1.5.1 Core Concepts

The central ideas of natural computation can be understood in terms of the composition of five basic mathematical and computational ideas: (1) fitness, (2) programs, (3) data, (4) dynamics, and (5) optimization.

• *Fitness.* The brain has to run without a *deus ex machina,* but nonetheless there must be a principle that produces the observed structures and behaviors. Such a principle is that of minimum description length, which captures the survival values of having good models.

• *Programs.* A fundamental concept is that of the brain's internal model of a computation. A very useful model contains two elements: *states* that represent abstractions of the current situation, and *operators,* or actions that govern transitions between states.

• *Data.* State spaces cannot be created out of whole cloth, but must be distilled from multidimensional sensory data. Continuous state spaces can be analyzed for useful structure in sensory data.

• *Dynamics.* A physical system's state has an associated *dynamics.* That is, the state vector changes in time. Such a trajectory is best described differentially in terms of the rate of change of the state vector with time. The easiest systems to deal with are *linear systems.* Such systems are completely characterized by their *eigenvalues* and *initial conditions.* Nonlinear systems can be approximated by linear systems local to stable equilibrium points, or *attractors.*

• *Optimization.* Given the shape of state spaces and the local evolution of trajectories in those spaces, one still needs a way of ranking the desirability of such spaces or trajectories. Ranking requires the ideas of optimization theory, which specifies ways of scoring state space trajectories as well as ways of using the scoring, or *objective function,* to find good paths.

1.5.2 Learning to React: Memories

Memories, or "look-up tables," are a way of storing precomputed correct reactions to situations. A stimulus pattern is sensed, and a response can be generated immediately. The main way of creating such associations is by learning from examples.

• *Content-addressable memory.* Content-addressable memory (CAM) is the principal kind of memory that the brain might have. Its central feature is that the content of a memory also serves as the address for its access. Such memories can be recovered even when they are distorted or have missing

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

parts. The job of a CAM is to fill in the missing parts appropriately. In simple CAMs all the neurons are stimulus driven.

• *Supervised learning.* Simple CAM memories may have trouble making fine distinctions. Remedies for this limitation are to add additional *internal* neurons and train the resultant network with a supervisory signal. The best of the supervised methods uses an error function between the classification produced by the network and the desired classification to improve the classification incrementally.

• *Unsupervised learning.* Supervised learning is an attractive model only in restricted circumstances. For most biological systems, unsupervised learning models that have internal neurons must also be used. These are predominantly stimulus driven, and the prototypes formed are functions of the data distribution of patterns.

1.5.3 Learning During a Lifetime: Programs

Look-up tables can be seen as defining primitive operations that can be composed to form larger behaviors. At the next level these neural primitives can be composed to direct the physical system in the course of more complex behaviors. Such compositions may be thought of as programs. A key feature of such programs is that initially the value of running the program cannot be assumed to be known at every step. Thus learning programs require dealing with reward that is *delayed.* To handle these delays, programs learn to form a *secondary reward function* that estimates future primary reward.

• *Markov systems.* The problem defined by any agent is not only to define a problem state space but also to define a model that expresses the transitions between different states. An extremely general way of performing this task that handles uncertainty is to allow probabilistic transitions between states. Such a system is called a *hidden Markov model.* The use of the word "hidden" signifies that the real states of the world are not known, and that the model states are therefore estimates.

• *Reinforcement learning.* Hidden Markov models (HMMs) provide the necessary substrate for describing reinforcement learning. Reinforcement learning associates rewards with transitions in HMMs such that paths through the HMM tend to maximize reward.

1.5.4 Learning Across Generations: Architectures

Memories and reinforcement strategies can reconfigure the existing structures but cannot alter the hardware design. For that purpose, genetic changes are needed. Genetic algorithms model the alteration of the genes during reproduction in order to create new architectural forms. These algorithms can be understood as experimenting with brain hardware. The basic combination of reproduction and fitness as measured by the organ-

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

isms' environment ensures that good features tend to spread rapidly throughout the population.

• *Genetic algorithms.* Genetic algorithms are a very abstract model of the process of sexual reproduction within a species' population in which strings of symbols represent the genetic code.

• *Genetic programming.* Biologically, DNA is only part of the story. Proteins must be manufactured that in turn assemble the body structures. In genetic algorithms this process is modeled with the fitness function, which implicitly scores the entire process, even though it is never explicitly represented. In contrast, genetic programs represent individuals as actual programs. The genetic operations are carried out on the actual program code. Consequently, an individual is a functioning program that can be directly tested in the environment.

1.6 THE GRAND CHALLENGE

At the beginning of the decade of the 1990s, scientists were asked to make a list of the problems that could be solved by spectacular advances in computing. Surprisingly, the challenge of understanding the brain with computational models is on the list only peripherally as "vision" and "speech." This reluctance to see computation as central justifiably reflects the immaturity of this endeavor. At this point computational models have not had the spectacular successes of the models of chemistry and physics. But the promise is there. To go further we must understand the computational framework with sufficient clarity in order to ask the right questions. The goal of this book is to further this process.

NOTES

1. From the introduction (by Eric Kandel) to Eric R. Kandel, James H. Schwartz, and Thomas M. Jessell, eds., *Principles of Neural Science,* 3rd ed. (Norwalk, CT: Appleton and Lange, 1991).

2. Patricia S. Churchland and Terrence J. Sejnowski, *The Computational Brain* (Cambridge, MA: MIT Press, 1992); Francis Crick, *The Astonishing Hypothesis: The Scientific Search for the Soul* (New York: Maxwell Macmillan International, 1994).

3. Roger Penrose, *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics* (Oxford: Oxford University Press, 1989).

4. D. A. Pomerleau, "Efficient Training of Artificial Neural Networks for Autonomous Navigation," *Neural Computation* 3 (1991):88–97.

5. G. Tesauro and T. J. Sejnowski, "A Parallel Network that Learns to Play Backgammon," *Artificial Intelligence Journal* 39 (1989):357–90.

6. Whitman Richards, ed., *Natural Computation* (Cambridge, MA: MIT Press, 1988).

7. Ibid., introduction.

8. Churchland and Sejnowski, *The Computational Brain.*

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

9. The figure is from *Review of Neuroscience* by Ben Pansky and Delmas J. Allen (New York: Macmillan, 1980). The designations of function in the caption are tentative, as the overall operation of the brain circuitry at this scale is far from determined; however, the descriptions are fairly commonly accepted (Kandel, Schwartz, and Jessell, *Principles of Neural Science*).

10. The figure is from David C. Van Essen, Charles H. Anderson, and Daniel J. Felleman, "Information Processing in the Primate Visual System: An Integrated Perspective," *Science* 255 (24 January 1992): 419–23.

11. The part of the figure showing the neural spike train is from B. W. Connors and M. J. Gutnick, "Intrinsic Firing Patterns of Diverse Neocortical Neurons," *Trends in Neurosciences* 13 (1990):98–99.

12. Compare axons to silicon gates, which normally connect to fewer than 10 other gates. Such connectivity would be impossible in silicon without dedicated hardware to specially manage the connections owing to the capacitance of the wires, but biological axons are specially designed to actively propagate the signal without loss.

13. Stephen Jay Gould, "The Basis of Creativity in Evolution" (presentation, Rochester Conference on Creation, University of Rochester, January 1987).

14. Churchland and Sejnowski, *The Computational Brain.*

15. A. N. Kolmogorov and V. A. Uspenskii, "Algorithms and Randomness," *Theory of Probabilities and its Applications* 32–33 (1987):425–55.

16. Jorma Rissanen, *Stochastic Complexity in Statistical Inquiry* (River Edge, NJ: World Scientific, 1989).

17. This example is from Ming Li and Paul Vitányi's standard text, *An Introduction to Kolmogorov Complexity and Its Applications* (New York: Springer-Verlag, 1993).

18. First suggested by Donald O. Hebb in *The Organization of Behavior: A Neuropsychological Theory* (New York: Science Editions, 1949).

19. In contrast, the chimpanzee brain is 80% of its final size at birth. Christopher Wills, *The Runaway Brain: The Evolution of Human Uniqueness* (New York: Basic Books, 1993).

20. Steven Rose, *The Making of Memory: From Molecules to Mind* (New York: Doubleday, Anchor Books, 1992).

21. This point is dramatically made in silicon circuitry in the form of special digital signal-processing chips that can process a video frame in just a fraction of the frame time. This ability has allowed the development of real-time computer vision algorithms for the first time.

22. The ideas in this section were originally developed by Allen Newell in his book *Unified Theories of Cognition* (Cambridge, MA: Harvard University Press, 1990).

23. Gregory L. Baker and Jerry P. Gollub, *Chaotic Dynamics: An Introduction* (New York: Cambridge University Press, 1990).

24. Newell, *Unified Theories of Cognition.*

25. Another reason that the modeling methods here are different from traditional symbol manipulation in AI is that the timescales are much shorter, too short to form a symbol.

EXERCISES

1. Comment on the use of simple models to describe complex structures, giving examples from different fields of cases where such models have proven useful and others where they have not.

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

2. Estimate the total length of axonal "wiring" in the brain in kilometers (a) assuming that the length of an axon is 0.1 mm, and (b) assuming the length of an axonal *branch* is .01 mm.

3. Derive a scheme to put Turing machines in correspondence with the integers. That is, given a Turing machine's state table, your job is to specify a unique integer that describes it.

4. It is known that humans can react to a visual input and push a button within 400 ms. Given that the visual input from the retina is carried by 1 million neurons, what does this suggest about the average complexity of the algorithm that the brain is using?

5. Given that the effects of incoming voltage signals take 10 ms to make a target cell fire, and that the signals travel at 1 meter/second, estimate the number of sequential synapses that could be involved in the 400-ms reaction of the previous problem.

6. A précis of Roger Penrose's book appears in *Behavioral and Brain Sciences* (Volume 13, 1990, pages 643–705), with commentaries on his position. After digesting the article and one or more of the commentaries, write your own commentary.

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025

Downloaded from http://direct.mit.edu/books/monograph/4666/bookpreview-pdf/2455500 by guest on 19 May 2025