CSLI Publications
Center for the Study of Language and Information
Leland Stanford Junior University
Printed in the United States
05 04 03 02 01 1 2 3 4 5

Library of Congress Cataloging-in-Publication Data

Foundations of real-world intelligence / edited by Yoshinori Uesaka, Pentti Kanerva, Hideki Asoh.
p. cm. -- (CSLI lecture notes ; no. 125)
Includes bibliographical references and index.
ISBN 1-57586-339-1 (acid-free paper) -- ISBN 1-57586-338-3 (pbk. : acid-free paper)
1. Artificial intelligence. 2. Neural networks (Computer science) 3. Evolutionary programming (Computer science) I. Uesaka, Yoshinori, 1936– . II. Kanerva, Pentti, 1937– . III. Asoh, Hideki, 1958– . IV. Series.
Q335 .F685 2001
006.3--dc21

2001043252

∞ The acid-free paper used in this book meets the minimum requirements of the American National Standard for Information Sciences—Permanence of Paper for Printed Library Materials, ANSI Z39.48-1984.

Please visit our web site at
http://cslipublications.stanford.edu/
for comments on this and other titles, as well as for changes and corrections by the author and publisher.

Contents

Preface **x**

General Introduction **1**
RWI Research Center, Electrotechnical Laboratory

1 Real-World Intelligence and the Real-World Computing Program **1**
Nobuyuki Otsu
1.1 Outline of the RWC Program 3
1.2 Real-World Intelligence 4
1.3 Concluding Remarks 8

2 Theoretical and Algorithmic Foundations of Real-World Intelligence **9**
Hideki Asoh
2.1 Objective 10
2.2 Approach 10
2.3 Research Issues 11
2.4 Organization of R&D and This Book 13
2.5 Concluding Remark 17

References **17**

I Inference and Learning with Graphical Models **19**
RWI Research Center, Electrotechnical Laboratory

3 An Overview of Theoretical Foundation Research in RWI Research Center **21**
Hideki Asoh, Kazuhisa Niki, Koiti Hasida, Shotaro Akaho, Masaru Tanaka, Yoichi Motomura, Tatsuya Niwa, and Kenji Fukumizu
3.1 Models and Algorithms 21
3.2 Frameworks of Learning 27

4 BAYONET: Bayesian Network on Neural Network 28
Yoichi Motomura
4.1 Bayesian Networks Based on Neural Networks 29
4.2 Implementation 32
4.3 Application 36
4.4 Conclusion 37

5 Multivariate Information Analysis 38
Kazuhisa Niki, Junpei Hatou, Toshiaki Kawamata, and Ikuo Tahara
5.1 Expression of Multivariate Analysis 38
5.2 Simulation 43
5.3 Structure Analyses of fMRI Data 49
5.4 Extended Functional Connectivity Analysis 51
5.5 Conclusion 54

6 Dialogue-based Map Learning in an Office Robot 55
Hideki Asoh, Yoichi Motomura, Toshihiro Matsui, Satoru Hayamizu, and Isao Hara
6.1 Dialogue-based Map Acquisition 55
6.2 System and Experiment 60
6.3 Discussion 64
6.4 Related Work 65
6.5 Conclusion and Future Work 66

7 Conclusion 68

References **68**

II Approximate Reasoning: Real-World Applications of Graphical Models 73
RWC Theoretical Foundation SNN Laboratory
Bert Kappen, Stan Gielen, Wim Wiegerinck, Ali Taylan Cemgil, Tom Heskes, Marcel Nijman, and Martijn Leisink

8 Mean Field Approximations 74
8.1 Mean Field Approximation with Structure 75
8.2 Boltzmann Machine Learning Using Mean Field Theory and Linear Response Correction 79
8.3 Second-order Approximations for Probability Models 85
8.4 Discussion 93

9 Medical Diagnosis 94
9.1 Probabilistic Modeling in the Medical Domain 95
9.2 Promedas, a Demonstration DSS 97
9.3 Discussion 99

10 Automatic Music Transcription 100
10.1 Dynamical Systems and the Kalman Filter 102
10.2 Tempogram Representation 106
10.3 Model Training 109
10.4 Evaluation 111
10.5 Discussion and Conclusions 116
References **119**

III Evolutionary Computation and Beyond 123
RWC Theoretical Foundation GMD Laboratory
Heinz Mühlenbein and Thilo Mahnig
11 Analysis of the Simple Genetic Algorithm 125
11.1 Definitions 125
11.2 Proportionate Selection 126
11.3 Recombination 126
11.4 Selection and Recombination 128
11.5 Schema Analysis Demystified 130
12 The Univariate Marginal Distribution Algorithm (UMDA) 133
12.1 Definition of UMDA 134
12.2 Computing the Average Fitness 137
13 The Science of Breeding 139
13.1 Single Trait Theory 140
13.2 Tournament Selection 144
13.3 Analytical Results for Linear Functions 145
13.4 Numerical Results for UMDA 147
13.5 Royal Road Function 147
13.6 Multimodal Functions Suited for UMDA Optimization 150
13.7 Deceptive Functions 151
13.8 Numerical Investigations of the Science of Breeding 152
14 Graphical Models and Optimization 154
14.1 Boltzmann Selection and Convergence 155
14.2 Factorization of the Distribution and the FDA 157
14.3 A New Annealing Schedule for the Boltzmann Distribution 159
14.4 Finite Populations 163
14.5 Population Size, Mutation, and Bayesian Prior 165
14.6 Constraint Optimization Problems 170
15 Computing a Bayesian Network from Data 171
15.1 LFDA—Learning a Bayesian Factorization 172
15.2 Optimization, Dependencies, and Search Distributions 175

16 System Dynamics Approach to Optimization 176
16.1 The Replicator Equation 176
16.2 Boltzmann Selection and the Replicator Equation 178
16.3 Some System Dynamics Equations for Optimization 179
16.4 Optimization of Binary Functions 181
17 Three Royal Roads to Optimization 184
18 Conclusion and Outlook 185
References **186**

IV Distributed and Active Learning 189
RWC Theoretical Foundation NEC Laboratory
19 Distributed Cooperative Bayesian Learning 190
Kenji Yamanishi
19.1 Introduction 190
19.2 Plain Model 192
20 Learning Specialist Decision Lists 209
Atsuyoshi Nakamura
20.1 Preliminaries 211
20.2 Algorithm S-Loss-Update 213
20.3 Algorithm SWML 218
20.4 Algorithm S-Fixed-Share-Update 222
21 The Lob-Pass Problem 226
Jun'ichi Takeuchi, Naoki Abe, and Shun'ichi Amari
21.1 Preliminaries 231
21.2 Upper Bounds on the Expected Regret 236
21.3 Concluding Remarks 248
References **248**

V Computing with Large Random Patterns 251
RWC Theoretical Foundation SICS Laboratory
Swedish Institute of Computer Science
22 Analogy as a Basis of Computation 254
Pentti Kanerva
22.1 Computer as a Brain and Brain as a Computer 256
22.2 Artificial Neural Nets as Biologically Motivated Models of Computing 257
22.3 Description vs. Explanation 258
22.4 The Brain as a Computer for Modeling the World, and Our Model of the Brain's Computing 259
22.5 Pattern Space of Representations 260
22.6 Simple Analogical Retrieval 265

22.7 Learning from Examples 266
22.8 Toward a New Model of Computing 269

23 The Sparchunk Code: A Method to Build Higher-level Structures in a Sparsely Encoded SDM 272
Gunnar Sjödin
23.1 Encoding Higher-Level Concepts 272
23.2 The SDM Model 273
23.3 Nonscaling for a Constant Error Probability ε 274
23.4 Sparse Coding 276
23.5 The Sparchunk Code 276
23.6 Clean-up of the Sparchunk Code 278
23.7 Summary 279
23.8 Appendix 279

24 Some Results on Activation and Scaling of Sparse Distributed Memory 283
Jan Kristoferson
24.1 Different Activation Probabilities for Writing and Reading? 283
24.2 Scaling Up the Memory 286

25 A Fast Activation Mechanism for the Kanerva SDM Memory 289
Roland Karlsson
25.1 The Jaeckel Selected-Coordinate Design 290
25.2 The New Selected-Coordinate Design 291
25.3 Results 292
25.4 Conclusions 293

26 From Words to Understanding 294
Jussi Karlgren and Magnus Sahlgren
26.1 The Meaning of 'Meaning' 294
26.2 A Case in Point: Information Access 295
26.3 Words as Content Indicators 297
26.4 Latent Semantic Analysis 299
26.5 Random Indexing 300
26.6 What Is Text, from the Perspective of Linguistics? 301
26.7 The TOEFL-Test 302
26.8 Experimental Set-Up 303
26.9 Results and Analysis 303
26.10 Some Cognitive Implications 305
26.11 Implications for Information Access 306
26.12 Meaning in Text 307

References **308**

Index 313

V

Computing with Large Random Patterns

RWC Theoretical Foundation SICS Laboratory
Swedish Institute of Computer Science

Pentti Kanerva, Gunnar Sjödin, Jan Kristoferson,
Roland Karlsson, Björn Levin, Anders Holst, Jussi Karlgren,
and Magnus Sahlgren

In this chapter we discuss a style of computing that differs from traditional numeric and symbolic computing and is suited for modeling neural networks. We focus on one aspect of "neurocomputing," namely, computing with large random patterns, or high-dimensional random vectors, and ask what kind of computing they perform and whether they can help us understand how the brain processes information and how the mind works.

The human mind remains a deep mystery to science. The issues are far-ranging, from philosophy to biology to engineering to everything human. It is not even obvious how it is possible to understand the mind scientifically, because understanding is itself a product of the mind—it presupposes a mind. However, we will not dwell here on the philosophical issues but simply assume that mind is a proper object of scientific investigation when understood in terms of the workings of the brain. More generally, the human mind is to be understood in terms of a nervous system that controls the body and is controlled by it, in an environment that includes many others of its kind, because that is the setting in which the mind has come into existence. We are therefore interested in

Foundations of Real-World Intelligence.
Yoshinori Uesaka et al. (eds.).

mechanisms that produce lifelike behavior, and we associate mind with the internal states of those mechanisms. Although the mechanisms are material, the task is to understand them in the abstract, i.e. in terms of their states.

The study of machines with states is part of the science of computing. The abstract model of traditional computing is the Turing Machine, consisting of a finite automaton with unlimited tape—the tape is a kind of working memory. A Turing Machine is defined completely by the initial contents of its tape together with a state-transition table that gives the output symbol, tape motion, and the next state as a function of the current state and input symbol.

The most compelling result of the Turing Machine concerns universality. It is not difficult to describe a Universal Turing Machine, i.e. one that can perform the function of any other Turing Machine. This involves encoding the state-transition table of one machine onto the tape of the other, the universal machine. The computer term for this is "emulation." Just about any computer can be made into a universal emulator, one that can emulate any other computer.

Some unfortunate conclusions have been drawn from this universality of computers: first that the brain cannot be more than a Universal Turing Machine; then that it too can be emulated by the universal emulator; and finally that our computers can do anything that a brain can. What this line of reasoning overlooks is that the brain must work in real time, whereas a Turing Machine needs only to accomplish its computations in a finite time. Therefore the Turing Machine is too abstract a model for brain's computing; it hardly helps us understand information precessing by the nervous system.

To highlight the essential abstractness of the Turing Machine, note that its definition is not at all concerned with the construction of the machine. It matters only that there be distinct states and symbols, and orderly state transitions based on them. In computing practice as well as in nervous systems, however, the actual mechanisms are crucial, for they constrain what can be done in a given time. Therefore, in trying to understand brains in computing terms, we are deeply concerned with their construction and must look at them with the eye of a computer engineer.

Construction, too, can be addressed at many levels. Most concrete is material construction. With brains this falls in the domain of neuroscience, and with computers in the domains of materials science and electronics. However, if the goal is to build computers with brainlike powers, the primary task is to understand the computing principles involved, which means understanding brains in terms of mathematical

models that allow the principles to be studied and tested.

When we describe a task for a computer, it is done in terms of functions, variables, and values. In the computer these are realized by *circuits*, and there are different kinds of circuits for different functions. Although the circuits are built of more basic electronic components, a computer-design engineer is concerned primarily with the circuits and only secondarily with the components. In other words, circuits are the lowest level for describing what a computer does, i.e. describing in computationally meaningful terms the computing task being performed. This suggests that brains, too, should be viewed at the circuit level in interpreting what they do, and that the meaningful states are the states of the circuits. Individual neurons make up the circuits, but in themselves they have very little meaning for understanding the mind.

The term neural circuit can be understood in several ways. Here we mean a circuit made of many essentially similar components, either neurons or mini-circuits, that operate in parallel and provide input to other circuits over large numbers of axons. Another common use of the term "neural circuit" describes situations like this: Excitatory neuron of type A is connected to neurons of type B and C that inhibit each other, and C also inhibits A. For us, this would be a component of a circuit, essential for its working but meaningless otherwise, and hence we call it a "mini-circuit."

Circuits in the brain are large; they are composed of thousands or millions of neurons. Their construction follows a general plan—certain kinds of neurons make certain kinds of connections—but no two brains are alike in their details. The simplest mathematical model of the brain's architecture maps a component of a circuit to a component of a high-dimensional vector, or of a large pattern, so that the entire pattern–vector represents the state of the circuit. Information representation of this kind is called *population coding* in neuroscience and *distributed representation* in cognitive science (connectionism) and in artificial neural nets. Note also that the terms "large pattern," "high-dimensional vector," and "point of a high-dimensional space" all mean the same thing. The size of the pattern is the same as the dimensionality of the space.

The states and symbols of a Turing Machine are defined as sets without further structure, so that two states (or symbols) are either the same or different, with nothing in between. All the "know-how" is in the state-transition table (and the initial tape). Similarly for pointers, with which traditional data structures and symbolic expressions are built: any two pointers are either the same or different—there are no degrees of similarity or difference—and all the structure is encoded by chaining pointers. When the machine's states are represented by points of a

high-dimensional space, however, a new possibility emerges: the computation can take advantage of the *algebraic* and *metric* properties of the space. It is this fact that makes computing with large patterns interesting and worthy of investigation and holds promise for explaining the computational powers of the brain—for making brains understandable to us.

In this chapter we use very simple spaces, starting with the binary, to illustrate the principal ideas of computing with large random patterns. Section 22 describes the binary spatter code and its use for modeling analogy. The powerful idea here is that of mapping between structures, and it relies on the algebra of the space. However, the simplest spaces have flaws that can be avoided in spaces with richer structure. Dense binary codes (with 0s and 1s equally likely) have a scaling problem that is avoided with *sparse* codes. which are explained and explored mathematically in sections 23–24. An item memory or "clean-up" memory is an essential part of any computing architecture based on large random patterns. Section 25 describes its efficient simulation on a conventional computer. The last section takes on challenges presented by language. First, sparse random vectors are used to capture meanings of words from text, and finally the ultimate challenge for brainlike computing is outlined, namely, dealing with the remarkable flexibility with which we use language.

Research into distributed representation spans several decades and is considerable. The ideas most germane to the ones presented here are Hinton's (1990) reduced representation, Smolensky's (1990) tensor-product variable binding, Pollack's (1990) recursive auto-associative memory (RAAM), Plate's (1994) holographic reduced representation (HRR), and Rachkovskij and Kussul's (2001) context-dependent thinning.

22 Analogy as a Basis of Computation

Pentti Kanerva

Analogy is a defining process of the human mind (Hofstadter, 1985; Gentner, Holyoak, and Kokinov, 2001). In this section we contrast the workings of the mind with traditional computing, particularly in dealing with analogy and language, and argue for the need for a new kind of computing that could support humanlike artificial intelligence. We then suggest that such computing can be performed with large random patterns (or high-dimensional random vectors; the words "pattern" and "vector" will be used synonymously) and describe operations on such patterns. We are led to a class of models foreseen by Hinton (1990) and pioneered by Plate (1994), called *holographic reduced representation*

(HRR). The ideas are illustrated with the simplest of such models, the binary *spatter code* (Kanerva, 1996), and several examples are worked out to show their relevance to analogy.

The human mind is unlike any computer or program we know. It is not literal, and when meaning is taken literally, the result can be funny or total nonsense. That's the humor of puns. This must mean that the human mind, although capable of being literal, is fundamentally figurative, or symbolical or analogical. How else could we judge a literal interpretation of a sentence as being at once both accurate and wrong?

The development of the human mind—our growing grasp of things—is largely the result of analogical perceiving and thinking. Some things are meaningful to us at birth or require no learning; they are mostly things necessary for survival. The rest we learn through experience. Some learning is associative, as when we learn cause and effect, and such learning prevails throughout the animal kingdom.

To follow an example, or imitate, is an advanced form of learning and is common in at least mammals and birds. It involves a basic form of analogy: The learner identifies with a role model—perceives itself as the other, makes an analogical connection or mapping between itself and the other.

Full-fledged analogy is central to human intelligence. We relate the unfamiliar to the familiar, and we see the new in terms of the old. This is most evident in language, which is thoroughly metaphorical. New and unfamiliar things are expressed and explained in familiar terms that are themselves understood not literally but figuratively. It is possible that full-fledged analogy and human language need each other, and our faculties for them may have coevolved.

Analogy is such an integral part of us that we hardly notice it or pay it its proper dues—until we try to program a computer to act in human ways, that is. AI has puzzled over the programming of humanlike behavior for nearly half a century. At first it was thought that programming computers to understand language—to translate, and the like—was just around the corner, waiting only for computers to get powerful enough. Now computers are powerful, many things have been tried and much has been learned, but the puzzle remains and we don't seem to have an idea of how to solve it. Apparently we must rethink computing and put figurative meaning and analogy at its center; we must find computing mechanisms that make these notions natural. This can be construed as designing a new kind of computer, a "cognitive computer," that is a better model of the brain than are present-day computers.

22.1 The Computer as a Brain and the Brain as a Computer

Equating computers with brains is itself an example of analogical thinking. Early computers were dubbed "electronic brains," computers have memory, and we even say that a program knows, wants, or believes so and so. Such anthropomorphizing seems natural to us and it serves a purpose. It brings a technological mystery within the realm of the familiar; we already have an idea of what the brain does, even if we don't know just how it does it, and speaking analogically expresses this.

We also talk of the brain as a computer. The appeal of doing so is that whereas the mechanisms of the brain are hidden, those of the computer are available to us; through them we might understand the brain's mechanisms. This is the principle behind Turing's imitation game in which a human interrogator tries to decide which of two respondents is human and which is a machine imitating a human. Accordingly, if we can build a machine that behaves in the same way as a natural system does, we have understood the natural system.

Analogies not only help our thinking but also channel and limit it. The computer analogy of the brain or of the mind has certainly done so, in that modeling in cognitive science and AI has been dominated by programs written for the computer, while philosophical and qualitative approaches have been looked on with suspicion.

Many things are modeled successfully on computers, such as weather, traffic flow, strength of materials and structures, industrial processes, and so forth. There are pitfalls, however, when the thing being modeled—the brain—is itself some kind of a computer. The danger is that our models begin to look like the computers they run on or the programming languages they are written in. For example, we talk of human short-term or working memory and think of the computer's active registers, or we talk of human long-term memory and think of the computer's permanent storage (RAM or disk), or we talk of the grammar of a language and think of a tree-structure or a set of rewriting rules programmed in Lisp. Of course these are analogical counterparts, but there is a danger of taking them too literally. Human memory works very differently than does computer memory, and the brain is not a Lisp machine nor the mind a logic program. Some analogical comparisons have not been at all useful in understanding how the mind works. Equating the brain with the computer's hardware and the mind with its software is an example. Finally, there is a worse danger of failing to notice what is missing in our models of the mind because it is missing or invisible in computers. To safeguard against this, we must treat the subject qualitatively: Our models may behave as we claim, but we must also ask, is that how peo-

ple behave? Is that, for example, how they use language? (See sec. 26 "From Words to Understanding.")

22.2 Artificial Neural Nets as Biologically Motivated Models of Computing

The computer's and brain's architectures are very different, and it is possible that the differences account for the difficulty of programming computers to be more lifelike and less literal-minded. This has motivated the study of alternative computing architectures called (artificial) neural nets (ANN), or parallel distributed processing (PDP), or connectionist architectures. We study these alternatives hoping that an architecture more similar to the brain's should produce behavior more similar to the brain's, which is a valid analogical argument. Unfortunately it does not tell us what in the architecture matters and what is incidental, and unfortunately meaning in our neural nets is hardly more figurative than in traditional computers.

However, neural-net research has made a valuable contribution by focusing on representation. Computer theoreticians and engineers know, for example, that the representation of numbers has a major effect on circuit design. A representation that works well for addition works reasonably well also for multiplication, whereas a representation that allows very fast multiplication is useless for addition. Thus a representation is a compromise that favors some operations and hinders others.

Information in computers is stored locally, that is, in data records composed of fields. Local representation—one unit per concept—is common also in artificial neural nets. The alternative is to distribute information from many sources over shared units. This is more brainlike, at least superficially, and it has been studied and has long been used with neural nets. Distributed representation appears to be fundamental to the brain's operation, and therefore a cognitive computer should be based on it. This means we must find out how to encode information into, and how to operate with, distributed representations.

Neural-net research has shown that distributed representations are robust and support some forms of generalization: representations (patterns) that are similar on the surface—close according to some metric—are treated similarly, for example as belonging to the same or similar classes. Distributed representations are also suitable for learning from examples. This type of learning takes place by statistical averaging or clustering of representations (self-organizing). It is not very creative but it can be subtle and lifelike, which makes it cognitively interesting. It can produce behavior that looks like rule-following although the system has no explicit rules, as Rumelhart and McClelland (1986) demonstrated

with the learning of the past tense of English verbs. This is a significant discovery, in that it demonstrates a principle that apparently governs the working of the mind in general and thus should govern the working of a cognitive computer. What we see and describe as rule-following is an emergent phenomenon that reflects an underlying mechanism; however, the rules need not produce the behavior even if they do accurately describe it.

22.3 Description vs. Explanation

The distinction between the description and explanation of behavior is so central that I will highlight it with an example. Consider heredity. Long before the genetic mechanisms of heredity were known, people knew about dominant and recessive traits and had figured out the basic laws of inheritance. For example, a plant species may come in three varieties, with white, pink, or red flowers, and cross-pollinating the white with the red always produces pink flowers. The specific rule is that all of the first generation is pink, and when pink-flowered plans are crossed with each other, one-fourth of the offspring is white, one-fourth red, and half pink. So we can say that the inheritance follows this rule. But no mechanism in the reproductive system keeps counting offspring to make sure that the proportions come out right, as if to say: "I have made so and so many white flowers, so it's time to make the same number of red flowers." It is not the rule that makes the proportions come out in a certain way; rather, the proportions are an outward reflection of the mechanism that passes traits from one generation to the next. It is significant, however, that long before chromosomes or genes or RNA and DNA were discovered, people speculated correctly about a hereditary mechanism that would produce offspring in those proportions. Clearly, the laws provided a useful description of the behavior, and accurate description can lead to discovery and explanation.

The situation is similar with regard to language and to mental functions at large. For example, we attribute the patterns of a language to its grammar and we devise sets of rules by which the grammar works. However, it is not the grammar that generates sentences in us when we speak or write. The regularities captured in the grammar are an outward expression of our underlying mechanisms for language—the grammar is an emergent phenomenon. This distinction is easily lost when we produce language output with computers, for there we actually use the grammar to generate sentences, and we work hard to develop a comprehensive grammar for a language. And when we think of the computer as a model of the brain and use computers to model mental functions, we tacitly assume that the brain uses grammatical rules to generate lan-

guage. Formal logic as a model of thinking can be criticized on similar grounds: It may describe rational thought but it does not explain rational thought. If taken in the right spirit, however, a proper description of thinking and language can help us discover the underlying mechanisms.

22.4 The Brain as a Computer for Modeling the World, and Our Model of the Brain's Computing

It is useful to think of the brain as a computer if we make the analogy between the two sufficiently abstract. But what in computers should we look at? Executing a sequence of programmed instructions for manipulating pieces of data stored in memory seems like an overly specified model of how the brain or the mind works. A more useful analogy is made at the level of computers as state machines, the states being realized as configurations of matter, or patterns in some physical medium. Mental states and subjective experience then correspond to—or are caused by—physical states so that when a physical state repeats, the corresponding subjective experience repeats. Thus the patterns that define the states are the objective counterpart of subjective experience. Our senses are the primary source of the patterns, and our built-in faculties for pleasure and pain give primary meaning to some of them. Brains are wired for rich feedback, and when the feedback works in such a way that an experience created by the senses—i.e. a succession of states—can later be created internally, we have the basis for learning. With learning, rich networks of meaningful states can be built.

The evolutionary function of this "computer" is to make the world predictable: The brain models the world as the world is presented to it by our senses. We are immersed in data received through multiple senses, and the sensory data defines our world, except that the real world cannot be fully sensed by any individual. Therefore, from the individual's point of view the world is *open-ended*.

Peripheral processing of sensory data is highly specialized data reduction refined by millions of years of evolution. Even with the reduction, the amount of data that we process further is huge compared to what we usually give our computer programs, and it is more like raw input than meaningful information or interpreted symbols. For example, our eyes do not pass letters and words on to the brain; rather, they pass patterns that the brain can interpret as letters and words.

Much of our sensory data is about the external environment, but an important part is internal. This internal component tells of our welfare, and its meaning is built in by evolution. We know at birth that pain is painful and should be avoided.

In addition to sensors, we interact with the environment with motors

that allow us to move and to affect the environment. The brain generates activity patterns in the motor neurons, and thus, computationally, the connection to the environment is in terms of activity *patters* over very large arrays of sensory and motor neurons. The function of the brain is to convert sensory patterns into motor patterns that promote the individual's welfare, which the brain measures by the internal sensory patterns with built-in meanings. This, then, is the computational setting for the development of human intelligence.

22.5 Pattern Space of Representations

Modeling the brain's computations in minute detail is neither possible nor necessary, because no two brains are identical. Instead, a model should capture some fundamental principle and allow it to be tested. The modeling itself is abstract, but when it corresponds to plausible neural mechanisms it can lead to an understanding of real brains and to the ability to build practical systems that employ the brain's processing principles.

The representation of information used in the model—that is, the entities or items with which the model computes—must reflect the brain's structures and mechanisms for collecting, combining, and transforming sensory data. This is overlooked in traditional AI, where the data structures are not correlated with brainlike mechanisms, and it has even been claimed that such correlating is wholly unnecessary (Fodor and Pylyshyn, 1988). In contrast, low-level representation—the physical configurations that carry the information—plays a central role in our modeling. It is governed by two ideas.

1. The representation is *uniform*: All things are represented by points of the same mathematical space, be they objects, properties, relations, functions, roles, fillers, composed structures, mappings of structures, and so on.
2. *Meaning* is internal and it *follows form*: Points close by in the representation space mean similar things—semantics cannot be separated from syntax.

This contrasts with traditional numeric and symbolic computing, which is mostly syntactic and takes meaning as supplied from the outside (the exception is some robots, where we are beginning to see examples of a system's own internal meaning).

A vector space is a particularly simple mathematical model that can satisfy the above two conditions, provided that its dimensionality N is high enough, say, several to ten thousands. Real vectors have been used for this kind of modeling by Plate (1994) and by Gayler and Wales

(1998); complex vectors have been used by Plate (1994) as well. The high dimensionality is more important than the nature of the dimensions, so that good modeling is possible even with binary dimensions (Kanerva, 1996; Rachkovskij and Kussul, 2001). Since the information capacity of one such binary vector is roughly equivalent to a page of text, every step of a computation can be thought of as turning a page.

We think of the system as attending to one N-dimensional pattern (N-vector) at a time and call it the system's attentional *focus* (Kanerva, 1988). The focus combines information from all the senses and for all the motors. Obviously it is possible to represent one thing very precisely in the focus, or to represent many things at once, each of them less precisely.

In traditional computing, a computer word or a database record is divided into fields that represent the parts of a whole that should be considered simultaneously and that make up a higher-level unit or concept. For example, a person could be identified by five attributes, name, sex, age, height, and marital status, so that the record would have these five fields. However, such fields are hard to justify in a brainlike computer, and therefore we make do without fields altogether. Instead, we combine N-vectors for the attributes into a single N-vector for their combination. This results in *distributed representation*, also called *holographic* or *holistic*. Operations on N-vectors for composing, decomposing, and mapping of representations are discussed next.

22.5.1 Operations and Their Algebra

We will write N-vectors (N-component patterns) in boldface letters and scalar quantities in italics. Thus v_n is the nth component of the N-vector $\mathbf{v}$. For matrixes we use bold uppercase Greek letters (e.g. $\mathbf{\Gamma}$). Specific examples of operations are given for binary vectors.

Computers have built-in operations (machine instructions) for combining and transforming data: for adding, subtracting, multiplying, and dividing two numbers, and for negating and shifting bits of a binary word, bitwise Boolean AND, OR, and XOR of two binary words, testing for a 0, and a few others. Similar operations are needed for computing with large patterns (with N-vectors). The following ones have been used for representing structure and for mapping of representations:

Unary Operations. A single pattern is transformed by scalar multiplication ($\mathbf{v} = k\mathbf{x}$, i.e. $v_n = kx_n$) without losing information, and it is transformed more generally by matrix multiplication ($\mathbf{v} = \mathbf{\Gamma x}$) with or without information loss, depending on the matrix $\mathbf{\Gamma}$. By "without information loss" we mean that the operation is invertible.

Unary operations on binary patterns include complementing and per-

muting of coordinates ($\mathbf{v} = \neg\mathbf{x}$ and $\mathbf{v} = \mathbf{\Pi x}$ where $\mathbf{\Pi}$ is a permutation matrix) without loss of information, and more generally multiplying with a matrix followed by thresholding ($\mathbf{v} = \langle \mathbf{\Gamma x} \rangle$), with loss of information.

Binary Operations. Two patterns are combined by coordinate-wise addition ($\mathbf{v} = \mathbf{x} + \mathbf{y}$) without loss of information, coordinate-wise multiplication ($v_n = x_n y_n$) with possible information loss, and more generally by compressing their outer-product matrix $\mathbf{x}\mathbf{y}^\mathsf{T}$ into an N-vector (notated with $\mathbf{v} = \mathbf{x} * \mathbf{y}$) with information loss.

The commonly used operations on binary patterns are coordinate-wise AND, OR, and Exclusive-OR ($\mathbf{v} = \mathbf{x} \wedge \mathbf{y}$, $\mathbf{v} = \mathbf{x} \vee \mathbf{y}$, and $\mathbf{v} = \mathbf{x} \otimes \mathbf{y}$), the first two of which lose information.

Ternary and Higher Operations. Three or more patterns are combined by coordinate-wise addition, possibly followed by scaling,

$$\mathbf{v} = \langle \mathbf{x} + \mathbf{y} + \cdots + \mathbf{z} \rangle$$

with information loss. This same notation also is used for the thresholded sum of binary patterns.

The algebraic properties of these operations govern composition and decomposition. *Distributivity* is important and it will be demonstrated below.

Typical of some operations is that they lose information, so that computing with them gives approximate results and iterated computing diverges. To counter the tendency to diverge, clean-up is required. We can think of it as being done by an *item memory* or a *clean-up memory* that stores all valid patterns—ones that the system knows—and retrieves the best-matching known pattern when cued with an approximate one, or retrieves nothing if the best match is no better than what results from random chance. The clean-up is reliable only if the dimensionality is in the thousands, which is the reason for requiring that the patterns be truly high-dimensional.

22.5.2 Composition and Decomposition, or Synthesis and Analysis

The following examples are based on traditional symbol processing, using roles and fillers, or variables and values, to compose information. Since it is most unlikely that the brain's computing would be organized in such a categorical fashion, we are merely demonstrating the power of these operations to combine syntax and semantics, or form and meaning, in a uniform representation. Our conjecture is that such combining is necessary if a system is to give its own internal meaning to things sensed about the environment.

We will demonstrate composition and decomposition with the very

simplest of patterns—namely, patterns with binary components (binary vectors). If the variables x, y, z have values a, b, c, respectively, their conjunction

$$(x = a)\ \&\ (y = b)\ \&\ (z = c)$$

can be encoded as follows. First, the three variables and the three values are represented by independent random 10,000-bit vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{a}, \mathbf{b}, \mathbf{c}$, with independent and identically distributed components that are 1s with probability 1/2 (the vectors are *dense*). The conjunction can then be encoded by the 10,000-bit vector sum

$$\mathbf{s} = \mathbf{x} \otimes \mathbf{a} + \mathbf{y} \otimes \mathbf{b} + \mathbf{z} \otimes \mathbf{c}$$

thresholded at 1.5: $\mathbf{v} = \langle \mathbf{s} \rangle$. This shows two levels of composition. One operation (XOR) is used to associate or *bind* each variable to a value, and another operation (normalized sum) is used to *merge* the three bound pairs into a single vector. Merging has also been called "super(im)posing," "bundling," and "chunking."

The following algebraic properties allow a composed pattern to be analyzed or decoded into its constituents:

1. binding is invertible (at least approximately) and the inverse is called the "unbinding" operator;
2. binding and unbinding distribute over merging (at least approximately);
3. the binding of two patterns produces a pattern that is *dissimilar* to both; and
4. the merging of patterns produces a pattern that is *similar* to the merged patterns.

The XOR is its own inverse so that given the bound pair $\mathbf{u} = \mathbf{x} \otimes \mathbf{a}$, XORing it with either yields the other; for example, $\mathbf{x} \otimes \mathbf{u} = \mathbf{x} \otimes (\mathbf{x} \otimes \mathbf{a}) = (\mathbf{x} \otimes \mathbf{x}) \otimes \mathbf{a} = \mathbf{a}$ finds the pattern to which $\mathbf{x}$ is bound in $\mathbf{v}$ (it finds the value of x in $\mathbf{u}$).

Decoding the composed pattern $\mathbf{v} = \langle \mathbf{x} \otimes \mathbf{a} + \mathbf{y} \otimes \mathbf{b} + \mathbf{z} \otimes \mathbf{c} \rangle$ relies on distributivity. For example, to find the value of x, we XOR $\mathbf{v}$ with $\mathbf{x}$ and get

$$\begin{aligned} \mathbf{a}' &= \mathbf{x} \otimes \mathbf{v} \\ &= \mathbf{x} \otimes \langle \mathbf{x} \otimes \mathbf{a} + \mathbf{y} \otimes \mathbf{b} + \mathbf{z} \otimes \mathbf{c} \rangle \\ &= \langle \mathbf{x} \otimes \mathbf{x} \otimes \mathbf{a} + \mathbf{x} \otimes \mathbf{y} \otimes \mathbf{b} + \mathbf{x} \otimes \mathbf{z} \otimes \mathbf{c} \rangle \end{aligned} \tag{22.1}$$

because XOR distributes over the thresholded sum of an odd number k of Boolean variables threshold at $k/2$ (Kanerva, 1996). This reduces to

$$\mathbf{a}' = \langle \mathbf{a} + \mathbf{x} \otimes \mathbf{y} \otimes \mathbf{b} + \mathbf{x} \otimes \mathbf{z} \otimes \mathbf{c} \rangle,$$

which is similar to $\mathbf{a}$ (see item 4 above); it is also similar to $\mathbf{x} \otimes \mathbf{y} \otimes \mathbf{b}$ and to $\mathbf{x} \otimes \mathbf{z} \otimes \mathbf{c}$, but they are not stored in the item memory and thus act as random noise.

22.5.3 Holistic Mapping

Mappings between composed patterns can be done in a like manner. We will demonstrate it with a mapping vector $\mathbf{M}$ that performs multiple substitutions at once, i.e. that substitutes the values d, e, f for a, b, c, respectively: $a \to d$, $b \to e$, and $c \to f$. It is given by

$$\mathbf{M} = \langle \mathbf{a} \otimes \mathbf{d} + \mathbf{b} \otimes \mathbf{e} + \mathbf{c} \otimes \mathbf{f} \rangle,$$

and it maps the pattern $\mathbf{v}$ as follows:

$$\mathbf{v} \otimes \mathbf{M} = \mathbf{w}' \approx \mathbf{w},$$

where $\mathbf{w} = \langle \mathbf{x} \otimes \mathbf{d} + \mathbf{y} \otimes \mathbf{e} + \mathbf{z} \otimes \mathbf{f} \rangle$ represents

$$(x = d)\ \&\ (y = e)\ \&\ (z = f),$$

which is the result of the three substitutions. The derivation is an exercise in distributivity and is similar to (22.1) above:

$$\begin{aligned}
\mathbf{v} \otimes \mathbf{M} &= \langle \mathbf{x} \otimes \mathbf{a} + \mathbf{y} \otimes \mathbf{b} + \mathbf{z} \otimes \mathbf{c} \rangle \otimes \langle \mathbf{a} \otimes \mathbf{d} + \mathbf{b} \otimes \mathbf{e} + \mathbf{c} \otimes \mathbf{f} \rangle \\
&= \langle \mathbf{x} \otimes \mathbf{a} \otimes \langle \mathbf{a} \otimes \mathbf{d} + \mathbf{b} \otimes \mathbf{e} + \mathbf{c} \otimes \mathbf{f} \rangle \\
&\quad + \mathbf{y} \otimes \mathbf{b} \otimes \langle \mathbf{a} \otimes \mathbf{d} + \mathbf{b} \otimes \mathbf{e} + \mathbf{c} \otimes \mathbf{f} \rangle \\
&\quad + \mathbf{z} \otimes \mathbf{c} \otimes \langle \mathbf{a} \otimes \mathbf{d} + \mathbf{b} \otimes \mathbf{e} + \mathbf{c} \otimes \mathbf{f} \rangle \rangle \\
&= \langle \langle \mathbf{x} \otimes \mathbf{a} \otimes \mathbf{a} \otimes \mathbf{d} + \mathbf{x} \otimes \mathbf{a} \otimes \mathbf{b} \otimes \mathbf{e} + \mathbf{x} \otimes \mathbf{a} \otimes \mathbf{c} \otimes \mathbf{f} \rangle \\
&\quad + \langle \mathbf{y} \otimes \mathbf{b} \otimes \mathbf{a} \otimes \mathbf{d} + \mathbf{y} \otimes \mathbf{b} \otimes \mathbf{b} \otimes \mathbf{e} + \mathbf{y} \otimes \mathbf{b} \otimes \mathbf{c} \otimes \mathbf{f} \rangle \\
&\quad + \langle \mathbf{z} \otimes \mathbf{c} \otimes \mathbf{a} \otimes \mathbf{d} + \mathbf{z} \otimes \mathbf{c} \otimes \mathbf{b} \otimes \mathbf{e} + \mathbf{z} \otimes \mathbf{c} \otimes \mathbf{c} \otimes \mathbf{f} \rangle \rangle \\
&= \langle \langle \mathbf{x} \otimes \mathbf{d} + \mathbf{x} \otimes \mathbf{a} \otimes \mathbf{b} \otimes \mathbf{e} + \mathbf{x} \otimes \mathbf{a} \otimes \mathbf{c} \otimes \mathbf{f} \rangle \\
&\quad + \langle \mathbf{y} \otimes \mathbf{b} \otimes \mathbf{a} \otimes \mathbf{d} + \mathbf{y} \otimes \mathbf{e} + \mathbf{y} \otimes \mathbf{b} \otimes \mathbf{c} \otimes \mathbf{f} \rangle \\
&\quad + \langle \mathbf{z} \otimes \mathbf{c} \otimes \mathbf{a} \otimes \mathbf{d} + \mathbf{z} \otimes \mathbf{c} \otimes \mathbf{b} \otimes \mathbf{e} + \mathbf{z} \otimes \mathbf{f} \rangle \rangle \\
&= \langle \langle \mathbf{x} \otimes \mathbf{d} + \text{noise}_1 + \text{noise}_2 \rangle \\
&\quad + \langle \text{noise}_3 + \mathbf{y} \otimes \mathbf{e} + \text{noise}_4 \rangle \\
&\quad + \langle \text{noise}_5 + \text{noise}_6 + \mathbf{z} \otimes \mathbf{f} \rangle \rangle \qquad (22.2) \\
&= \mathbf{w}' \\
&\approx \langle \mathbf{x} \otimes \mathbf{d} + \mathbf{y} \otimes \mathbf{e} + \mathbf{z} \otimes \mathbf{f} \rangle \\
&= \mathbf{w}.
\end{aligned}$$

The pattern $\mathbf{w}'$ is approximately correct; it is close enough for identifying $\mathbf{w}$ in the clean-up memory. The exact result is obtained by mapping (XORing) $\mathbf{v}$ with $\mathbf{M}_1 = \mathbf{v} \otimes \mathbf{w}$, so that $\mathbf{M}_1 \approx \mathbf{M}$. Notice that the mapping $\mathbf{M}$ performs the three substitutions simultaneously in any context,

not just when the variables x, y, z are involved, and that a specific *example* of the three substitutions, such as $\mathbf{v} \rightarrow \mathbf{w}$, yields an approximate mapping $\mathbf{M}_1$ that also performs the substitutions in any context.

22.6 Simple Analogical Retrieval

Holistic mapping is a remarkable operation that has no obvious counterpart in traditional symbol processing. It is suggestive of analogy and could therefore be a mechanism for higher intelligence. The following example deals with figurative meaning and analogy. See also Eliasmith and Thagard (2001).

Let **F** be a pattern representing France: that its capital is Paris, its geographic location is Western Europe, and its monetary unit is the franc. Denote the patterns for capital, Paris, geographic location, Western Europe, money, and franc by **ca**, **Pa**, **ge**, **We**, **mo**, and **fr**. France is then represented by the pattern

$$\mathbf{F} = \langle \mathbf{ca} \otimes \mathbf{Pa} + \mathbf{ge} \otimes \mathbf{We} + \mathbf{mo} \otimes \mathbf{fr} \rangle.$$

Probing **F** for "the Paris of France" is done by mapping (XORing) it with **Pa**. The derivation is analogous to (22.1) and it yields

$$\mathbf{F} \otimes \mathbf{Pa} = \langle \mathbf{ca} + \mathbf{ge} \otimes \mathbf{We} \otimes \mathbf{Pa} + \mathbf{mo} \otimes \mathbf{fr} \otimes \mathbf{Pa} \rangle$$

and is approximately equal to **ca**:

$$\mathbf{F} \otimes \mathbf{a} \approx \mathbf{ca}.$$

Thus XORing with **Pa** has mapped **F** approximately into **ca**, meaning that Paris is France's capital. Here the meaning of Paris is literal.

Much more than that can be done in a single mapping operation. Let **S** be a pattern for Sweden with capital Stockholm (**St**), located in Scandinavia (**Sc**), and with monetary unit krona (**kr**). This information about Sweden is then represented by the pattern

$$\mathbf{S} = \langle \mathbf{ca} \otimes \mathbf{St} + \mathbf{ge} \otimes \mathbf{Sc} + \mathbf{mo} \otimes \mathbf{kr} \rangle.$$

We can now ask "What is the Paris of Sweden?" If we take the question literally and do the mapping $\mathbf{S} \otimes \mathbf{Pa}$, as above, we get nothing recognizable, so we must take Paris in a more general, figurative sense. "The Paris of France" produced a recognizable result above (i.e. approximately **ca**, meaning capital), so we can use it: we can map **S** with $\mathbf{F} \otimes \mathbf{Pa}$ and get $\mathbf{S} \otimes \mathbf{F} \otimes \mathbf{Pa}$, which is recognizable as the pattern for Stockholm:

$$\mathbf{S} \otimes \mathbf{F} \otimes \mathbf{Pa} \approx \mathbf{St}.$$

The derivation is similar to (22.1). The significant thing in $\mathbf{S} \otimes \mathbf{F} \otimes \mathbf{Pa}$ is that $\mathbf{S} \otimes \mathbf{F}$ can be thought of as a *binding of two composed patterns of equal status* (Sweden and France), rather than a binding of a variable

to a value, and it can also be thought of as a holistic mapping $\mathbf{M}_1$ (see sec. 22.5.3) between France and Sweden, capable of answering analogical questions of the kind "What is the Paris of Sweden?" and "What is the krona of France?"

Finally, the case of multiple substitutions as discussed in section 22.5.3 on "Holistic Mapping": What will happen to the pattern for France if we substitute Stockholm for Paris, Scandinavia for Western Europe, and krona for franc, all at once, and how is the substitution done? We create a mapping vector $\mathbf{M}$ by binding the corresponding items to each other and by merging the results:

$$\mathbf{M} = \langle \mathbf{Pa} \otimes \mathbf{St} + \mathbf{We} \otimes \mathbf{Sc} + \mathbf{fr} \otimes \mathbf{kr} \rangle.$$

Mapping the pattern for France with $\mathbf{M}$ is an example of (22.2) and yields

$$\mathbf{F} \otimes \mathbf{M} \;\approx\; \langle \mathbf{ca} \otimes \mathbf{St} + \mathbf{ge} \otimes \mathbf{Sc} + \mathbf{mo} \otimes \mathbf{kr} \rangle \;=\; \mathbf{S},$$

so that a single mapping operation composed of *multiple substitutions* changes the pattern for France to an approximate pattern for Sweden, recognizable by the clean-up memory.

22.7 Learning from Examples

We have just seen an example of a mapping vector ($\mathbf{M}$) that performs several substitutions at once. It is composed of individual substitutions (each substitution appears as a bound pair, which are then merged), and the mapping is between things that share structure (same roles, different objects). We will now do the reverse: We map between structures that share objects (same objects in two different relations). The mappings are constructed from examples, so that this demonstrates analogical inference. It is an example in miniature of the more ambitious Copycat program of Hofstadter (1984) and Mitchell (1993).

We will look at two relations, one of which implies the other: "If x is the mother of y, then x is the parent of y," represented symbolically by $m(x, y) \rightarrow p(x, y)$. We take a specific example $m(A, B)$ of the mother-of relation and compare it to the corresponding parent-of relation $p(A, B)$, to get a mapping M_1 between the two. We then use this mapping on another pair (U, V) for which the mother-of relation holds, to see whether M_1 maps $m(U, V)$ into the corresponding parent-of relation $p(U, V)$.

We encode "A is the mother of B," or $m(A, B)$, with the random N-vector $\mathbf{mAB} = \langle \mathbf{m} + \mathbf{m}_1 \otimes \mathbf{A} + \mathbf{m}_2 \otimes \mathbf{B} \rangle$, where $\mathbf{m}$ encodes (names) the relation and $\mathbf{m}_1$ and $\mathbf{m}_2$ encode its two roles. Similarly, we encode "A is the parent of B," or $p(A, B)$, with $\mathbf{pAB} = \langle \mathbf{p} + \mathbf{p}_1 \otimes \mathbf{A} + \mathbf{p}_2 \otimes \mathbf{B} \rangle$. Then

$$\mathbf{M}_1 \;=\; \mathbf{M}_{AB} \;=\; \mathbf{mAB} \otimes \mathbf{pAB} \tag{22.3}$$

maps a specific instance of the mother-of relation into the corresponding instance of the parent-of relation, because $\mathbf{mAB} \otimes \mathbf{M}_1 = \mathbf{mAB} \otimes \mathbf{M}_{AB} = \mathbf{mAB} \otimes (\mathbf{mAB} \otimes \mathbf{pAB}) = \mathbf{pAB}$.

The mapping $\mathbf{M}_{AB}$ is based on one example; so we must ask, is it possible to generalize based on only one example? When the mapping is applied to another instance $m(U, V)$ of the mother-of relation, which is encoded by $\mathbf{mUV} = \langle \mathbf{m} + \mathbf{m}_1 \otimes \mathbf{U} + \mathbf{m}_2 \otimes \mathbf{V} \rangle$, we get the pattern $\mathbf{W}$:

$$\mathbf{W} = \mathbf{mUV} \otimes \mathbf{M}_{AB}.$$

Does $\mathbf{W}$ resemble $\mathbf{pUV}$?

We will measure the similarity of patterns by their correlation ρ (i.e. by normalized covariance; $-1 \leq \rho \leq 1$). The correlations reported here are exact: They are mathematical mean values or expectations (the means are over complete sets of composed patterns—complete in the sense that they include all possible bit combinations of their component patterns, so that if the composed patterns involve a total of b "base" vectors, all patterns will be 2^b bits).

If we start with randomly selected (base) vectors $\mathbf{m}$, $\mathbf{m}_1$, $\mathbf{m}_2$, $\mathbf{p}$, $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{A}$, $\mathbf{B}, \ldots, \mathbf{U}$, $\mathbf{V}$ that are pairwise uncorrelated ($\rho = 0$), we observe first that $\mathbf{mAB}$ and $\mathbf{pAB}$ are uncorrelated but that $\mathbf{mAB}$ and $\mathbf{mUV}$ are correlated because they both include $\mathbf{m}$ in their composition; in fact, $\rho(\mathbf{mAB}, \mathbf{mUV}) = 0.25$ and, similarly, $\rho(\mathbf{pAB}, \mathbf{pUV}) = 0.25$. When $\mathbf{W}$ is compared to $\mathbf{pUV}$ and to other vectors, there is a tie for the best match: $\rho(\mathbf{W}, \mathbf{pUV}) = \rho(\mathbf{W}, \mathbf{pAB}) = 0.25$. All other correlations with $\mathbf{W}$ are lower: with the related (reversed) parent-of relations $\mathbf{pBA}$ and $\mathbf{pVU}$ it is 0.125, with an unrelated parent-of relation $\mathbf{pXY}$ it is 0.0625, and with $\mathbf{A}$, $\mathbf{B}, \ldots, \mathbf{U}$, $\mathbf{V}$, $\mathbf{mAB}$, and $\mathbf{mUV}$ it is 0. So based on only one example, $m(A, B) \rightarrow p(A, B)$, it cannot be decided whether $m(U, V)$ should be mapped to the original "answer" $p(A, B)$ or should generalize to $p(U, V)$.

Let us now look at generalization from three examples of the mother-of implying the parent-of relation: What is $m(U, V)$ mapped to by $\mathbf{M}_3$ that is based on $m(A, B) \rightarrow p(A, B)$, $m(B, C) \rightarrow p(B, C)$, and $m(C, D) \rightarrow p(C, D)$? This time we will use a mapping vector $\mathbf{M}_3$ that is the sum of three binary vectors,

$$\mathbf{M}_3 = \mathbf{M}_{AB} + \mathbf{M}_{BC} + \mathbf{M}_{CD},$$

where $\mathbf{M}_{AB}$ is as above and $\mathbf{M}_{BC}$ and $\mathbf{M}_{CD}$ are defined similarly. Since $\mathbf{M}_3$ itself is not binary, mapping $\mathbf{mAB}$ or $\mathbf{mUV}$ with $\mathbf{M}_3$ cannot be done with an XOR. However, we can use an equivalent system in which binary patterns are bipolar, by replacing 0s and 1s with 1s and -1s, and bitwise XOR $\otimes$ with coordinate-wise multiplication $\times$. Then the

mapping can be done with multiplication, the patterns can be compared with correlation, and the results obtained with $\mathbf{M}_1$ still hold. Notice that now $\mathbf{M}_{AB} = \mathbf{mAB}\times\mathbf{pAB}$, for example (cf. eqn. (22.3)).

Mapping with $\mathbf{M}_3$ gives the following results. To check whether it works at all, consider $\mathbf{W}_{AB} = \mathbf{mAB}\times\mathbf{M}_3$. It is most similar to **pAB** ($\rho = 0.71$), as expected, because $\mathbf{M}_3$ contains $\mathbf{M}_{AB}$. Its other significant correlations are with **mAB** (0.41) and with **pUV** and **pVU** (0.18). Thus the mapping $\mathbf{M}_3$ strongly supports $m(A,B) \rightarrow p(A,B)$. It also unambiguously supports the generalization $m(U,V) \rightarrow p(U,V)$, as seen by comparing $\mathbf{W}_{UV} = \mathbf{mUV}\times\mathbf{M}_3$ with **pUV**. The correlation is $\rho(\mathbf{W}_{UV}, \mathbf{pUV}) = 0.35$; the other significant correlations of $\mathbf{W}_{UV}$ are with **pAB** and **pVU** (0.18) and with **pBA** (0.15) because they all include the pattern **p** (parent-of).

To track the trend further, we look at generalization from five examples, $m(A,B) \rightarrow p(A,B), m(B,C) \rightarrow p(B,C), \ldots, m(E,F) \rightarrow p(E,F)$, giving the mapping vector

$$\mathbf{M}_5 = \mathbf{M}_{AB} + \mathbf{M}_{BC} + \mathbf{M}_{CD} + \mathbf{M}_{DE} + \mathbf{M}_{EF}.$$

Applying it to **mAB** yields $\rho(\mathbf{mAB}\times\mathbf{M}_5, \mathbf{pAB}) = 0.63$ (the other correlations are lower, as they were for $\mathbf{M}_3$), and applying it to **mUV** yields $\rho(\mathbf{mUV}\times\mathbf{M}_5, \mathbf{pUV}) = 0.40$ (again the other correlations are lower).

When the individual mappings $\mathbf{M}_{XY}$ are analyzed, each is seen to contain the three *kernel vectors* $\mathbf{m}\times\mathbf{p}$, $\mathbf{m}_1\times\mathbf{p}_1$, and $\mathbf{m}_2\times\mathbf{p}_2$, plus other vectors that act as noise and average out as more and more of the mappings $\mathbf{M}_{XY}$ are added together. The analysis is analogous to (22.2) and yields:

$$\begin{aligned}
\mathbf{M}_{XY} &= \mathbf{mXY}\times\mathbf{pXY} \\
&= \langle \mathbf{m} + \mathbf{m}_1\times\mathbf{X} + \mathbf{m}_2\times\mathbf{Y}\rangle\times\langle \mathbf{p} + \mathbf{p}_1\times\mathbf{X} + \mathbf{p}_2\times\mathbf{Y}\rangle \\
&= \langle\langle \mathbf{m}\times\mathbf{p} + \text{noise}_0\rangle + \langle \mathbf{m}_1\times\mathbf{p}_1 + \text{noise}_1\rangle \\
&\quad + \langle \mathbf{m}_2\times\mathbf{p}_2 + \text{noise}_2\rangle\rangle.
\end{aligned}$$

The three kernel vectors are responsible for the generalization, and from them we can construct a *kernel mapping* from mother-of relation to the parent-of relation:

$$\mathbf{M}^* = \mathbf{m}\times\mathbf{p} + \mathbf{m}_1\times\mathbf{p}_1 + \mathbf{m}_2\times\mathbf{p}_2.$$

When it is used to map **mUV**, we get a maximum correlation with **pUV**, as expected, i.e. $\rho(\mathbf{mUV}\times\mathbf{M}^*, \mathbf{pUV}) = 0.43$; correlations with other parent-of relations are $\rho(\mathbf{mUV}\times\mathbf{M}^*, \mathbf{pXY}) = 0.14$ ($\mathbf{X} \neq \mathbf{U},\ \mathbf{Y} \neq \mathbf{V}$) and 0 with everything else.

The results are summarized in figure 22.1, which relates the amount of data to the strength of inference and generalization. The data are

examples or instances of the mother-of relation implying the parent-of relation, $m(x, y) \rightarrow p(x, y)$, and the task is to map either an old (fig. 22.1a) or a new (fig. 22.1b) instance of mother-of into parent-of. The data are taken into account by encoding them into the mapping vector $\mathbf{M}_k$.

Figure 22.1a shows the effect of new data on old examples. Adding examples into the mapping makes it less specific, and consequently the correlation for old inferences (i.e. with **pAB**) decreases, but it decreases also for all incorrect alternatives. Figure 22.1b shows the effect of data on generalization. When the mapping is based on only one example, generalization is inconclusive ($\mathbf{mUV}\times\mathbf{M}_1$ is equally close to **pAB** and **pUV**), but when it is based on three examples, generalization is clear, as $\mathbf{M}_3$ maps **mUV** much more closely to **pUV** than to any of the others. Finally, the kernel mapping $\mathbf{M}^*$ represents a very large number of examples, a limit as the number of examples approaches infinity, at which point the correct inference is the clear winner.

22.8 Toward a New Model of Computing

Analogical mapping between two information structures was first modeled with the methods of traditional symbolic processing (e.g. Gentner, 1983), and analogy appears to be fundamentally symbolical (Gentner and Markman, 1993). The simplest form of holographic reduced representation, the binary spatter code, has been used here to demonstrate it in a setting more appropriate for neural nets. Similar demonstrations have been made by Chalmers (1990) and others (e.g. Boden and Niklasson, 1995) using Pollack's (1990) RAAM, and by Plate (1994) using real-vector HRR. The lesson from such demonstrations is that certain kinds of representations and operations on them make it possible to perform symbolic tasks with distributed representations suitable for neural nets. Furthermore, when patterns are used as if they were symbols (Gayler and Wales, 1998), we do not need to configure different neural nets for different data structures. A general-purpose net that operates with such patterns is thus somewhat like a general-purpose computer that runs programs for a variety of tasks.

Our two examples use a traditional symbolic setting with roles and fillers, an operation for binding the two, and another operation for combining bound pairs (and singular identifiers) into representations for new, higher-level compound entities such as relations. The point, however, is not to develop a neural-net Lisp machine but to see what additional properties a system based on distributed representation might have, and whether they might lead to a new model of computing and to improved modeling of high-level mental functions.

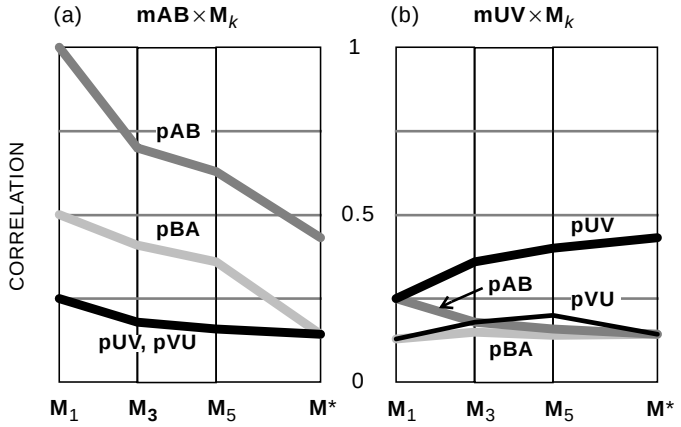

FIGURE 22.1. Leaning from example: The mappings $\mathbf{M}_k$ map the mother-of relation to the parent-of relation. They are computed from k specific instances or examples of **mXY** being mapped to **pXY**. Each map $\mathbf{M}_k$ includes the instance "**mAB** maps to **pAB**" and excludes the instance "**mUV** maps to **pUV**." The mappings are tested by mapping an "old" instance **mAB** (a) and a "new" instance **mUV** (b) of mother-of to the corresponding parent-of. The graphs show the closeness of the mapped result (its correlation ρ) to different alternatives for parent-of. The mapping $\mathbf{M}^*$ is a kernel map that corresponds to an infinite number of examples. Graph b shows good generalization (**mUV** is mapped closest to **pUV**) and discrimination (it is mapped much further from, e.g. **pAB**) based on only three examples ($\mathbf{M}_3$). The thickness of the heavy lines corresponds to slightly more than ± 1 standard deviation around the expected correlation ρ when $N = 10{,}000$.

The binding and merging operators that are used to encode traditional data structures can also be used to encode mappings between structures, such as the kernel map between two relations. Furthermore, individual instances or examples of the mapping $\mathbf{M}$ are encoded with the binding operator—by binding corresponding instances of the two relations—and several such examples can be merged into an estimate of the kernel map by simply adding them together or averaging.

That the averaging of vectors for structured entities is meaningful is a consequence of the kind of representation used. It has no counterpart in traditional symbolic representation and is, in fact, one of the most exciting and tantalizing aspects of distributed representation. Another is the "holistic" mapping of structures with mapping vectors such as $\mathbf{M}$. This type of mapping suggests the possibility of learning structured rep-

resentations from examples without using explicitly encoded roles and fillers or relying on high-level rules. This would be somewhat like how humans learn. For example, children can learn to use language grammatically without explicit instruction in grammar rules, just as they pick up the sound patterns of their local dialect without being told about vowel shifts and so on. To model this kind of learning, we would still use the binding and merging operators and possibly one or two additional operators, but instead of binding the objects of a new instance to abstract roles, as was done in the examples of this section, we would bind them to the objects of an old instance—an example, or prototype—so that the example, rather than an abstract frame consisting of roles, would serve as the template. Averaging over many such examples could then produce representations from which the abstract notions of role and filler and other high-level concepts could be inferred. In essence, some traditions of AI and computational linguistics would be overturned. Instead of basing our systems on abstract structures such as grammars, we would base them on examples and would discover grammars in the representations that the system produces by virtue of its mechanisms. The rules of grammar would then reflect underlying mental mechanisms the way that the rules of heredity reflect underlying genetic mechanisms; and it is just the underlying mechanisms that interest us and that we want to understand.

Correlations obtained with HRRs tend to be low. Even the highest correlation for generalization in this section is only 0.43 and it corresponds to the kernel map (fig. 22.1b); we usually have to distinguish between two quite low correlations, such as 0.3 and 0.25. Such fine discrimination is impossible if the patterns have only a few components, but when they have several thousand, even small correlations and small differences in them are statistically significant. This explains the need for very high-dimensional patterns. For example, with $N = 10{,}000$, correlations have a standard deviation no greater than ± 0.01, so that the difference between 0.3 and 0.25 is highly significant (more than 5 standard deviations). These statistics suggest that $N = 10{,}000$ allows very large systems to work reliably and that $N = 100{,}000$ would usually be unnecessarily large. Both high-dimensionality and randomness are essential properties of the representation, and they are what make it rather brainlike. What is appealing about large random patterns is that they have rich and subtle mathematical properties, and they lend themselves to parallel computing.

For a computer to work like the human mind, it must be extremely flexible in its use of symbols. It cannot stumble over the multiplicity of meanings that a word can have; rather, it must be able to benefit from

that multiplicity. The human mind conquers the unknown by making analogies to that which is known; it understands the new in terms of the old. In so doing it creates ambiguity or, rather, it creates rich networks of mental connections and becomes robust. That is the exact opposite of how we view ambiguity in traditional computing. Our challenge then is to find the right kinds of representations, ones that allow meaningful new "symbols" to be created by simple operations on existing patterns. This is the prime motive for the study of distributed representations based on large random patterns.

23 The Sparchunk Code: A Method to Build Higher-level Structures in a Sparsely Encoded SDM[†]

Gunnar Sjödin

23.1 Encoding Higher-Level Concepts

Traditionally, structured information has been represented in the form of records with fields. However, the architecture of the brain suggests that it uses distributed representations: to exaggerate somewhat, "everything" is stored "everywhere." This is the approach of Plate's (1994) holographic reduced representation (HRR) for real or complex vectors and Kanerva's (1996) spatter code for *dense* binary vectors, i.e. for vectors with approximately as many 0s as 1s. The task in these approaches is to build robust representations of higher-level concepts out of lower-level ones and then to be able to analyze the composite higher-level representation by taking it apart again into its constituents.

By abuse of language, we will in the following use "concept" to mean "representation of concept." Concepts are to be thought of as vectors with binary, real, or complex values. Suppose for instance that $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$ are three concepts, all vectors of the same length, and we want to merge them into a higher-level concept $\mathbf{Y}$, also a vector of the same length. This is done with different techniques for HRR and the spatter code by forming a new vector in a robust way from the $\mathbf{X}_i$s. The HRR uses averaging of vectors and the spatter code uses bitwise summing followed by thresholding to yield a mean vector. Robustness here means that if we change the parts, $\mathbf{X}_i$, a little the $\mathbf{Y}$ vector will also change just a little. The process of overlaying concepts like this will in the following be called *chunking*.

The analysis, i.e. decoding or breaking a composite concept down into its constituents, is divided into two stages. First we *probe* the compound concept to get approximations of its parts. After that we need to *clean up* these approximations to get exact copies of the constituents. For the probing the HRR uses a correlation operator (Plate, 1994), whereas the spatter code makes use of the XOR operator (Kanerva, 1996).

In this paper we are interested in binary representations, specifically in their use together with Kanerva's (1988) sparse distributed memory model (SDM). It turns out that the spatter code cannot be cleaned up using the SDM model. The problem with the clean-up arises in the following way: The $\mathbf{Y}$ probed for, e.g. $\mathbf{X}_1$, could be viewed as a noisy copy of $\mathbf{X}_1$. The relative Hamming distance will in this case be 0.25. This means that they are close in Hamming space. However, they are not close enough for us to use the probed $\mathbf{Y}$ as an address approximation of $\mathbf{X}_1$ given to an SDM memory to retrieve from this memory what we have stored in $\mathbf{X}_1$, in this case $\mathbf{X}_1$ itself.

To remedy this problem we turn our attention to *sparse codes*. A sparse pattern, used as an address to SDM, may be considered a small subset of a set equal in size to the address length. In this interpretation an occurrence of a 1 indicates membership in the small subset. We will define a chunking operation for sparse codes via *thinning*, *translation*, and *ORing*. This is what we will call the *sparchunk code*. Chunking two sparse subsets in this way gives a subset of the same size as the constituents and with an overlap with each of these of roughly half the size of the subsets. The probing is done with translation.

A main point of this paper is that we will be able to use a variant of the SDM model, the Jaeckel (1989b) hyperplane design, to do the clean-up for the sparchunk code. In demonstrating the feasibility of this we will also show that the *load* of the memory defined as the proportion between the number of stored patterns and the number of storage vectors can be allowed to approach infinity as the memory grows. There is no contradiction in this with information theory. The reason is that a sparseness of codes implies that the entropy of a code is low; the information given by each holder of a 0 or a 1 is very low so that the information-theoretic content supplied by the memory in a reading operation is much lower than in the case of dense codes.

23.2 The SDM Model

The Kanerva sparse distributed memory (SDM) is a memory with a set of M storage locations of U-dimensional vectors of integers. The memory is addressed within a binary address space of dimension N, where $M \ll 2^N$. For relevant literature on SDM see Jaeckel (1989a,

1989b), Kanerva (1988, 1992, 1993), Kristoferson (1995), and Sjödin (1995, 1996). When a U-dimensional binary vector $\mathbf{W}$ is stored *at* an address $\mathbf{X}$ a mechanism activates a small subset of the M locations and $\mathbf{W}$ is added by vector addition to the contents of these after first transforming 0s to -1s.

There is an interpretation of SDM as a model of the cerebellum (Kanerva, 1992). In this interpretation the storage locations correspond to the granule cells, the address is delivered to the memory via the mossy fibers, and the datum length U corresponds to the number of Purkinje cells.

In the original SDM model every location is associated with a fixed binary address chosen at random, and those locations whose associated addresses are at a sufficiently small Hamming distance from $\mathbf{X}$ are activated.

In the Jaeckel (1989a) selected-coordinate model, each location is associated with a *mask* of K selected coordinates, i.e. a set of indices between 1 and N. In the interpretation of SDM as a model for cerebellum K corresponds to the number of mossy-fiber synapses of a granule cell. Each of the selected coordinates is assigned either 0 or 1. If, for a given location, the assigned values in this mask coincide with the values of the corresponding bits of $\mathbf{X}$, the location is activated. In general an activation mechanism should have the *robustness* property: two addresses at small Hamming distances from each other should activate roughly the same set of locations.

When we later read the memory at an address $\mathbf{Y}$ close to $\mathbf{X}$, the same activation mechanism is used as when storing. In the original SDM model the sum of the activated locations is thresholded at 0 for each position in the datum; i.e. if a value is smaller than 0 it is transformed to 0 and otherwise to 1. This reading procedure can be improved considerably (Sjödin, 1995, 1996).

23.3 Nonscaling for a Constant Error Probability ε

Let $T+1$ be the number of stored data vectors (i.e. the size of the training set). The sum obtained at output position u when reading can be expressed as $S^{(u)} = L_0 B_0^{(u)} + \sum_1^T L_t B_t^{(u)}$ (Kanerva, 1993; Kristoferson, 1995). Here L_0 is the number of storage locations activated both when storing the vector $(B_0^{(u)})_{1 \le u \le U}$ at $\mathbf{X}$ and later when trying to recover it by reading at an address $\mathbf{Y}$ close to $\mathbf{X}$, and L_t is the number of locations activated both by $\mathbf{Y}$ and the storage operation of the vector $(B_t^{(u)})_{1 \le u \le U}$. We will consider one position at a time and then suppress the position indicator u. Denote the load of the memory, $\frac{T}{M}$,

by τ. Throughout we will use the activation probability $p = (2\tau M^2)^{-\frac{1}{3}}$, known to be the best for reading at a correct address (Kanerva, 1993; Kristoferson, 1995). Using other activation probabilities does not change anything in principle in this paper.

We say that we have a probability of error (in the address) equal to ε if for each address coordinate the probability of $\mathbf{X}$ and $\mathbf{Y}$ having different bit values in this coordinate is ε. It is furthermore required that these errors occur independently of each other.

The usual SDM model (Kanerva, 1993) does not scale with a constant probability of error ε: if we let M and T increase while ε remains constant, the probability of error in the read word B_0 will increase and actually approach $\frac{1}{2}$ for each position. In other words, the result will be just noise. This is true for both the basic model and the Jaeckel model.

Let us consider the Jaeckel model with mask lengths K. Consider a location activated by $\mathbf{X}$. To be activated also by $\mathbf{Y}$ each coordinate belonging to the mask must remain unchanged. The probability for each coordinate to remain correct is $1-\varepsilon$. Thus the probability of the location to be activated by $\mathbf{Y}$ is $(1-\varepsilon)^K$, so that the expected number of locations activated by both $\mathbf{X}$ and $\mathbf{Y}$ is

$$E(L_0) = (1-\varepsilon)^K \times (\text{number of locations activated by } \mathbf{X}).$$

This may also be expressed by saying that the signal in $S^{(u)}$ is reduced by a factor $(1-\varepsilon)^K$. Since the noise from other storage operations is roughly the same for $\mathbf{X}$ and $\mathbf{Y}$ we conclude that

$$\rho(\varepsilon, \tau) \quad \approx \quad (1-\varepsilon)^K \rho(0, \tau), \tag{23.1}$$

where $\rho(\varepsilon, \tau)$ is the signal-to-noise ratio for error probability ε and load τ. Now, let M increase and keep τ constant. Then, as is easily seen (Kanerva, 1993), $\rho(0, \tau)$ will converge to $\frac{1}{\sqrt{\tau}}$, and hence, since

$$p \;=\; 2^{-K} \;=\; (2\tau M^2)^{-\frac{1}{3}},$$

equation (23.1) reduces to 0.

Consider using the SDM as a clean-up memory for chunked dense codes. Let it have the size of the cerebellum, $M = 10^{10}$, and let $\tau = 0.1$. Then we get $K = 22$. Suppose that $\mathbf{Y}$ is the result of chunking $\mathbf{X}$ with two other patterns. Then $\varepsilon = 0.25$ and

$$(1-\varepsilon)^K = 0.75^{22} \approx \frac{1}{560},$$

so that the expected number of activated locations containing the information, i.e. those activated not only by $\mathbf{Y}$ but also by $\mathbf{X}$, is

$$\frac{1}{560} \cdot 2^{-K} M \;=\; \frac{1}{560} \cdot 2^{-22} \cdot 10^{10} \;\approx\; 4,$$

out of an expected number of totally 2384 activated locations.

We also get

$$\rho(0.25, 0.1) \approx 0.0056.$$

Thus the result will be just noise, which means that the the clean-up cannot be done in this way.

23.4 Sparse Coding

To remedy the above problem we keep K small (e.g. 3–6, as in the cerebellum) and fixed with growing M. To avoid activating too many locations, we use sparse coding, i.e. allow only addresses with a small number $a \ll N$ of 1s. We let a location be activated by the presence of 1s in the selected coordinates associated with the location. The model we then get is in fact exactly Jaeckel's (1989b) hyperplane design. The probability that an arbitrarily chosen address will activate a given location and that a randomly chosen location will be activated by a given address is

$$\begin{aligned} p &= \frac{\binom{N-K}{a-K}}{\binom{N}{a}} = \frac{\binom{a}{K}}{\binom{N}{K}} = \frac{a}{N} \cdot \frac{a-1}{N-1} \cdot \ \cdots \ \cdot \frac{a-K+1}{N-K+1} \\ &\approx \left(\frac{a}{N}\right)^K. \end{aligned}$$

From now on we will fix K at 3. Thus the proportion $\alpha = \frac{a}{N}$ of 1s is determined by $\alpha = p^{\frac{1}{3}}$.

23.5 The Sparchunk Code

Because of the nonscaling with constant ε, the ordinary Jaeckel model and the basic Hamming-distance model of SDM are unsuitable as a clean-up memory for the (dense) spatter code in Kanerva (1996). Here we discuss another version of sparse encoding of higher-level concepts, the sparchunk code, in which

- we only use one, noncommutative and nonassociative, binary operation;
- we construct sequences rather than sets; and
- all constituents, in particular variable names, of a compound word can be found. This can be done without prior knowledge of the structure of the compound word or of the names of the occurring variables.

In the following, we consider a sparse address $\mathbf{X}$ to be a subset of the integers between 0 and $N-1$ (inclusive), corresponding to the indices where $\mathbf{X}$ has 1s. Let the desired size of an address be a. Construct a *thinning* mechanism $\mathbf{X} \to \mathbf{X}'$, where $\mathbf{X}'$ has b elements. The size b is

chosen such that if $\mathbf{X}$ and $\mathbf{Y}$ are unrelated the expected size of $\mathbf{X}' \cup \mathbf{Y}'$ is a, i.e.

$$\left(1 - \frac{b}{N}\right)^2 = 1 - \frac{a}{N},$$

which gives

$$b = \frac{a}{1 + \sqrt{1 - \frac{a}{N}}}, \tag{23.2}$$

so that $b \approx \frac{a}{2}$ for small $\frac{a}{N}$, i.e. for a sparse address. Of course b has to be approximated to a natural number. For $\mathbf{X} = (x_0, \ldots, x_{N-1})$, let $\chi_r(\mathbf{X}) = (x_{-r}, \ldots, x_{N-1-r})$, where the indices of the variables are calculated mod N. Now define the *chunking* operation $*$ as

$$\mathbf{X} * \mathbf{Y} = \chi_1(\mathbf{X}') \cup \chi_2(\mathbf{Y}').$$

Note that $\chi_{-1}(\mathbf{X} * \mathbf{Y})$ is close to $\mathbf{X}$ but not to $\mathbf{Y}$, whereas $\chi_{-2}(\mathbf{X} * \mathbf{Y})$ is close to $\mathbf{Y}$ but not to $\mathbf{X}$. In fact, the proportional overlap between $\mathbf{X}$ and $\mathbf{X} * \mathbf{Y}$ is at least $\frac{b}{a} > \frac{1}{2}$. Multiple chunking will be replaced by

$$\mathbf{X}_1 * (\mathbf{X}_2 * (\mathbf{X}_3 * (\ldots * \mathbf{X}_n)) \ldots).$$

We can now build arbitrary recursive structures like

$$f_0(f_1(a_1, \ldots, a_{n_1}), \ldots, f_m(a_1, \ldots, a_{n_m})),$$

where the f_is are the first items in the chunked words. The chunking operator is robust provided that the thinning operator is; i.e. if $\mathbf{X}_1 \approx \mathbf{Y}_1$ and $\mathbf{X}_2 \approx \mathbf{Y}_2$ then $\mathbf{X}_1 * \mathbf{X}_2 \approx \mathbf{Y}_1 * \mathbf{Y}_2$.

There are many variations on this theme. The thinning mechanism could be used after the union operation. There are some drawbacks and some advantages with doing so. The thinning mechanism can be defined in a host of ways but it must remain deterministic and robust. One way of defining it is as follows. Let the desired number of 1s in $\mathbf{X}'$ be c, where $c = b$ in the situation above and $c = a$ if we do the thinning after the ORing. Let $f : \{0,1\}^N \to R$, where almost any function will do. Now define $\mathbf{X}'$ as

$\mathbf{X}'(i)$	=	1	if $x_i = 1$ and $f(\chi_i(\mathbf{X}))$ is among the c largest $f(\chi_j(\mathbf{X}))$s among those j for which $x_j = 1$. If there is a problem with ties, this is resolved, e.g. by letting low indices have priority.
$\mathbf{X}'(i)$	=	0	otherwise.

Note that

$$\chi_r(\mathbf{X})' = \chi_r(\mathbf{X}'),$$

which makes this method suitable also for the case when thinning is done after the ORing. As a simple example of how this works consider the following. Let $N = 10$ and $a = 3$ and let

$$\begin{aligned} f(x_0, \ldots, x_9) &= \sum_{i=0}^{9} x_i \cdot i^2 \\ \mathbf{X} &= 0100100100 \\ \mathbf{Y} &= 1001010000. \end{aligned}$$

Consider the case where thinning is done before ORing. The formula for b gives 1.63, which is approximated to 2. Thus $c = b = 2$. We get

$$\begin{aligned} f(\chi_1(\mathbf{X}')) &= 2^2 + 5^2 + 8^2 = 93 \\ f(\chi_4(\mathbf{X}')) &= 1^2 + 5^2 + 8^2 = 90 \\ f(\chi_7(\mathbf{X}')) &= 1^2 + 4^2 + 8^2 = 81 \end{aligned}$$

and

$$\begin{aligned} f(\chi_0(\mathbf{Y}')) &= 0^2 + 3^2 + 5^2 = 34 \\ f(\chi_3(\mathbf{Y}')) &= 3^2 + 6^2 + 8^2 = 109 \\ f(\chi_5(\mathbf{Y}')) &= 0^2 + 5^2 + 8^2 = 89 \end{aligned}$$

and hence

$$\begin{aligned} \mathbf{X}' &= 0100100000 \\ \mathbf{Y}' &= 0001010000, \end{aligned}$$

so that

$$\begin{aligned} \chi_1(\mathbf{X}') &= 0010010000 \\ \chi_2(\mathbf{Y}') &= 0000010100 \end{aligned}$$

and

$$\mathbf{X} * \mathbf{Y} = 0010010100.$$

23.6 Clean-up of the Sparchunk Code

The problem we face is to retrieve, or "clean up," $\mathbf{X}_1$ and $\mathbf{X}_2$ from $\mathbf{X}_1 * \mathbf{X}_2$. Let the proportional overlap between the address $\mathbf{X}$ and the address $\mathbf{Y}$ be $\delta \geq \delta_0 > 0$, where δ_0 is known. In the clean-up case we have at least the overlap given by (23.2), i.e. $\delta_0 \geq \frac{1}{2}$. We will use the memory described in section 23.4 auto-associatively, i.e. $U = N$, to store the addresses $\mathbf{X}_1, \mathbf{X}_2, \ldots$ themselves. These are patterns that the system "knows about" and is expected to recall. In reading the memory, the sums L_0, L_t, and B_t are distributed as $\mathrm{Bin}(M, \delta^3 p)$, $\mathrm{Bin}(M, p^2)$, and $\mathrm{Bin}(1, \alpha)$, respectively. B_0 is the value we want as the result of our reading. Let us use the threshold $A = \frac{\delta_0^3}{2} pM + p^2 MT\alpha$ for deciding

whether to read a 0 or a 1. Note that $E(L_0) = \delta^3 pM \geq \delta_0^3 pM$. This threshold makes the probability of error about the same for the two cases if $\delta \approx \delta_0$.

Admittedly, the straightforward application of chunking results in clusters. There are methods to handle this problem, which we do not have the space to discuss in this paper. Thus we will assume that the stored addresses are stochastically independent.

The following theorem shows that, in contrast with the dense model, the sparse model can retrieve data with arbitrarily good exactness if we let the size of the memory grow. This is so even if we let the load to approach infinity. In the case where the load is constant ($t = 0$), the theorem says that if we let U approach infinity faster than $M^{\frac{2}{9}}$ then the retrieval becomes arbitrarily exact. Its precise formulation is given as a "convergence in probability" result, when the size of the memory grows.

Theorem 23.1. *Let* $0 \leq t < \frac{1}{4}$ *and let* $\lim_{M\to\infty} UM^{-\frac{2+t}{9}} = \infty$. *Then, for* $\tau \sim M^t$ *and a positive* ε,

$$\lim_{M\to\infty} P(\text{number of errors} > \varepsilon a) \quad = \quad 0.$$

The proof is given in the appendix below. Note that the methods used in Sjödin (1996) require U to be at least $\approx M^{\frac{1}{3}}$ and that in the cerebellum $U \approx M^{\frac{2}{3}}$. The above theorem directly implies that we may let $\tau \to \infty$ with M and still reach convergence in probability. In fact, τ may be allowed to grow as a positive power of M.

23.7 Summary

We have demonstrated how to form representations of higher-level composite concepts from lower-level sparse ones, keeping the same sparseness in the composite pattern. Used as patterns for the sparse SDM model (the Jaeckel hyperplane design) we get a system where this model can be used as a clean-up memory, i.e. used for finding the constituent parts of a composite pattern. With increasing memory the clean-up can be made arbitrarily exact.

23.8 Appendix

This appendix gives a proof of theorem 23.1. In order not to get too involved in technical details, the proof is given for the case where τ is constant. The general proof is analogous. First we need some general results.

23.8.1 Large Deviations

From the Tchebycheff inequality we get lemma 1 (see Durrett, 1995, p. 38).

Lemma 1. *Let S_n be any sequence of random variables, and let b_n be any sequence such that* $\lim_{n\to\infty} \frac{b_n^2}{\sigma^2(S_n)} = \infty$. *Then*

$$\lim_{n\to\infty} P\left(|S_n - E(S_n)| > b_n\right) \quad = \quad 0.$$

Lemma 2. *Let X have distribution* $\mathrm{Bin}(n,p)$. *Then, for any $c \geq 0$,*

$$P\left(|X - np| > c\right) \quad \leq \quad 2e^{-\frac{2c^2}{n}}, \quad 0 \leq p \leq 1 \tag{23.3}$$

$$P\left(|X - np| > c\right) \quad \leq \quad 2e^{-\frac{c^2}{4np}}, \quad \frac{c}{n} \leq p \leq \frac{1}{4}. \tag{23.4}$$

Proof. Let $c = nb$. We may assume that $X = \sum_1^n X_i$, where the X_is are independent with distribution $\mathrm{Bin}(1,p)$. The usual Tchebycheff inequality trick in large deviation theory yields

$$P(X - np \leq -nb) \quad \leq \quad e^{-ng(b)}$$

where

$$\begin{aligned} g(b) \quad &= \quad \max\left(b\zeta - \log E(e^{-\zeta(X_1 - p)})\right) \\ &= \quad (p-b)\log\frac{p-b}{p} + (q+b)\log\frac{q+b}{q}. \end{aligned}$$

We have

$$\begin{aligned} g(0) \quad &= \quad g'(0) = 0 \\ g''(b) \quad &= \quad \frac{1}{(p-b)(q+b)} \\ g^{(3)}(b) \quad &\leq \quad 0 \ for\ b \leq \frac{p-q}{2}. \end{aligned}$$

Thus $g''(b) \geq 4$, $g''(b) \geq \frac{1}{pq}$ if $p \geq q$, and $g''(b) \geq \frac{1}{2q}$ if $q \leq \frac{1}{4}$ and $b \leq q$. Integrating these inequalities twice and dualizing with respect to p and q gives us equations (23.3) and (23.4). □

23.8.2 Proof of the Theorem in the Special Case Where τ is Constant

In reading from memory, let

$$\begin{aligned} Z^{(u)} \quad &= \quad S^{(u)} - \theta \\ &= \quad L_0 B_0^{(u)} + \sum_1^T L_t B_t^{(u)} - \ p^2 MT\alpha - \frac{\delta_0^3 pM}{2}. \end{aligned}$$

Note that L_0 and L_t are the same for all positions u, whereas the

$B_t^{(u)}$s are independent Bin$(1,\alpha)$-distributed variables. L_0 is Bin$(M,\delta^3 p)$-distributed and the L_ts are Bin(M,p^2)-distributed. We will suppress the position-index u. Let

$$\begin{array}{rcll} L'_t & = & L_t & \text{if } L_t \le 1 \\ L''_t & = & 1 & \text{if } L_t = 2 \text{ and } \quad 0 \text{ otherwise} \\ \tilde{L}_t & = & L_t & \text{if } L_t \ge 3 \text{ and } \quad 0 \text{ otherwise} \end{array}$$

so that $L_t = L'_t + 2L''_t + \tilde{L}_t$. Here L'_t is distributed as Bin$(1, Mp^2(1-p^2)^{M-1})$ and L''_t as Bin$(1, \binom{M}{2}p^4(1-p^2)^{M-2})$. The L_0 and L_ts are independent for different ts and hence so are the L'_ts, L''_ts, and $\tilde{L}_t$s. Rewrite Z as follows

$$\begin{aligned} Z &= \sum_1^T L'_t(B_t-\alpha) + 2\sum_1^T L''_t(B_t-\alpha) \\ &\quad + \sum_1^T \tilde{L}_t(B_t-\alpha) + \left(\sum_1^T L_t - p^2MT\right)\alpha \\ &\quad + L_0B_0 - \frac{\delta_0^3 pM}{2}. \end{aligned}$$

Let

$$\begin{aligned} \mathcal{A} &= \left\{L_0 \ge \tfrac{3}{4}\delta_0^3 pM\right\} \\ &\quad \cap \left\{\left|\sum_1^T L_t - p^2MT\right| \alpha \le \tfrac{1}{16}\delta_0^3 pM\right\} \\ &\quad \cap \left\{\sum_1^T \tilde{L}_t \le \tfrac{1}{16}\delta_0^3 pM\right\} \\ &\quad \cap \left\{\left|\sum_1^T L'_t - p^2MT\right| \le \tfrac{1}{2}p^2MT\right\} \\ &\quad \cap \left\{\left|\sum_1^T L''_t - \frac{1}{2}p^4M^2T\right| \le \tfrac{1}{4}p^4M^2T\right\}. \end{aligned}$$

Let $\mathcal{A}'$ be the logical complement of $\mathcal{A}$. Using lemma 1, it is straightforward to show that

$$\lim_{M\to\infty} P\left(\mathcal{A}'\right) = 0.$$

It remains to show that

$$\lim_{M\to\infty} P\left(\text{number of errors} > \varepsilon a | \mathcal{A}\right) = 0.$$

It is sufficient to show that $P(\text{number of errors} > \varepsilon a)$ conditioned on a setup of L_0, L_ts satisfying $\mathcal{A}$ converges to 0 uniformly in these L_0, L_ts. Thus assume that we have such a setup with fixed L_0 and L_ts. Then $\sum_1^T L'_t B_t$ is $\text{Bin}(\sum_1^T L'_t, \alpha)$–distributed and similarly for $\sum_1^T L''_t B_t$. Furthermore,

$$\sum_1^T L'_t(B_t^{(u)} - \alpha) + 2\sum_1^T L''_t(B_t^{(u)} - \alpha)$$

are independent for different indices u. Regardless of whether the correct value B_0 is 0 or 1, an error can occur only if

$$Z \;=\; \sum_1^T L'_t(B_t - \alpha) + 2\sum_1^T L''_t(B_t - \alpha) \geq \frac{\delta_0^3}{8} pM.$$

The now-independent error probabilities in each position satisfy

$$\begin{aligned} P(\text{error}) \;&\leq\; P\left(\left|\sum_1^T L'_t(B_t - \alpha)\right| > \tfrac{1}{16}pM\right) \\ &\quad + P\left(\left|\sum_1^T L''_t(B_t - \alpha)\right| > \tfrac{1}{32}pM\right). \end{aligned}$$

Hence, using (23.4) and (23.3) in lemma 2, we can find suitable positive constants C and D such that

$$P(\text{error}) \leq e^{-CM^{\frac{2}{9}}} + e^{-DM^{\frac{1}{3}}} \leq \frac{\varepsilon}{2}\alpha$$

for M large since $\alpha \sim M^{-\frac{2}{9}}$. So now the errors in different positions are distributed independently as $\text{Bin}(1, \tilde{p_u})$, where $\tilde{p_u} < \frac{\varepsilon\alpha}{2}$. Let X be $\text{Bin}(U, \frac{\varepsilon\alpha}{2})$–distributed. By (23.4) in lemma 2,

$$\begin{aligned} P(\text{number of errors} > \varepsilon a) \;&\leq\; P\left(X - \frac{\varepsilon\alpha}{2} > \frac{\varepsilon\alpha}{2}\right) \\ &\leq\; e^{-\frac{\varepsilon\alpha U}{8}} \;\to\; 0 \end{aligned}$$

if $\alpha U \to \infty$, i.e. if $M^{-\frac{2}{9}}U \to \infty$. □

Comment. We have implicitly assumed that the L_ts are i.i.d. binomially distributed. This is approximately true for memories with a long address length N and is assumed implicitly elsewhere in the literature. It has been indirectly verified by simulations (Manevitz and Nati, University of Haifa, personal communication).

24 Some Results on Activation and Scaling of Sparse Distributed Memory[†]

Jan Kristoferson

In computing with large patterns, some operations produce approximate items that need to be "cleaned up" (i.e. identified with their exact counterparts). This is done with a clean-up memory that stores all valid vectors known to the system and retrieves the best-matching vector when activated by a noisy vector. Kanerva's sparse distributed memory (SDM) is a natural candidate for such a clean-up memory. Let us consider an SDM of Jaeckel's (1989a) selected-coordinates design. We first give a brief explanation of the concept.

An *address* is a binary string (vector) of 1s and 0s. A *datum* is a binary string (vector) of 1s and −1s. Let N be the length (dimension) of the addresses and U the length (dimension) of the data. A *hard location* is a place in the memory where data are stored. The *content* of a hard location is a U-dimensional vector of integers. The coordinates of the content vectors are called *positions*. In conventional neural-net terms, hard locations are hidden units, their input weights define the position, and their output weights define the contents. A hard location is given by a *mask* (a subset of K coordinates out of the N coordinates in the address strings) and there is one bit for each coordinate in the mask, so in total there are $\binom{N}{K}2^K$ possible hard locations. An address $\mathbf{X}$ *activates* a hard location h if all bits in $\mathbf{X}$ lying in h's mask match the corresponding bits in h. *Storing a datum at the address* $\mathbf{X}$ means adding (as a vector) the datum to the contents of all hard locations activated by $\mathbf{X}$. *Reading at the address* $\mathbf{X}$ means calculating by some method, from the contents of the hard locations activated by $\mathbf{X}$, a datum to be read at $\mathbf{X}$. Here we consider just the standard reading method: We sum the contents and choose, for each position, the reading 1 if the sum is ≥ 0, and -1 if the sum is < 0. Let M be the number of hard locations, and $p = 2^{-K}$ the *probability of activation*.[1]

24.1 Different Activation Probabilities for Writing and Reading?

There has been some debate in the literature whether it would be helpful to use different activation probabilities p and r for writing and reading in

[1]The probability that a randomly chosen hard location is activated by a randomly chosen address.

the memory, respectively. In particular, it has been suggested by Kanerva (1993) that it might be reasonable to take $r > p$ in the situation where many noisy copies of each datum are written at noisy addresses.

Let us fix a model for this situation. Let us say that a binary vector $\mathbf{A}$ is ϵ-*disturbed* from another binary vector $\mathbf{B}$ if, for every bit in $\mathbf{A}$ independently, the probability of this bit being different from the corresponding bit in $\mathbf{B}$ is equal to ϵ. To begin with, assume $r \geq p$ and consider the following stochastic model.

(a) Choose randomly (and independently) the "ideal" addresses $\mathbf{X}_t$, $0 \leq t \leq T$.

(b) Choose $\mathbf{Y}$ and $\mathbf{X}_{t,k}$, $0 \leq t \leq T, 1 \leq k \leq n$ independently, with $\mathbf{Y}$ ϵ-disturbed from $\mathbf{X}_0$ and $\mathbf{X}_{t,k}$ η-disturbed from $\mathbf{X}_t$.

(c) Choose the M hard locations randomly, where each hard location is given by a mask with K bits. The writing activation is as usual, with probability $p = 2^{-K}$. For the reading activation a submask of the given (writing) mask with $L \leq K$ bits is chosen randomly for each hard location, giving the activation probability $r = 2^{-L} \geq p$.

As usual, we consider the signal-to-noise ratio ρ (see below) for a single content position. For this position, we take the following steps.

(d) Choose randomly the "ideal" datum bits B_t, $1 \leq t \leq T$. It is assumed that the B_ts are independent and also independent of all activation figures, i.e. stochastic variables ($L_{t,k,h}$ below) determined by which hard locations are activated by which addresses. The bit B_0 to be retrieved from the memory is assumed to be 1.

(e) Choose $B_{t,k}$, $0 \leq t \leq T, 1 \leq k \leq n$ randomly and independently, with $B_{t,k}$ β-disturbed from B_t. The $B_{t,k}$s will then also be independent of all activation figures. $B_{t,k}$ is written at $\mathbf{X}_{t,k}$,[2] $0 \leq t \leq T, 1 \leq k \leq n$.

Consider the stochastic variables $L_{t,k,h}$ defined thus: $L_{t,k,h} = 1$ if hard location h is activated by both $\mathbf{Y}$ and $\mathbf{X}_{t,k}$,[3] and $= 0$ otherwise. The sum read at the considered position is then

$$Z = \sum_{t=0}^{T} \sum_{k=1}^{n} \sum_{h=1}^{M} L_{t,k,h} B_{t,k}. \tag{24.1}$$

Granted that Z is approximately normally distributed (this assumption will be discussed below), the probability of getting the wrong reading at the given position is approximately $\Phi(-\rho)$, where Φ is the standard normal distribution function and $\rho = \frac{E(Z)}{\sigma(Z)}$ (the signal-to-noise

[2] with activation probability p.

[3] with activation probabilities r and p, respectively.

ratio). We find (Kristoferson, 1997) that

$$\rho^2 \approx f(p,r) = \frac{\text{num}}{\text{denom}} \tag{24.2}$$

where

$$\begin{aligned} \text{num} &= nMpr^{\gamma}(1-2\beta)^2 \\ \text{denom} &= 1 + (n-1)(1-2\beta)^2 p^{\lambda} r^{\kappa-\lambda-\gamma} \\ &\quad + [M - 1 + (1-2\beta)^2(1-n-M)]pr^{\gamma} \\ &\quad + Tr^{1-\gamma}[1 + (n-1)(1-2\beta)^2 p^{\lambda} \\ &\quad + (M - 1 + (1-2\beta)^2(n-1)(M-1))pr], \end{aligned}$$

where $\lambda = -\log_2[(1-\eta)^2 + \eta^2]$, $\kappa = -\log_2[(1-\epsilon)(1-\eta)^2 + \epsilon\eta^2]$ and $\gamma = -\log_2[(1-\epsilon)(1-\eta) + \epsilon\eta]$.

Now fix any value of r between 0 and 1. Then $f(p,r)$ is a function of p, where $0 < p \leq r$. If we divide num and denom by p, we get a constant numerator and a denominator of the form $ap^{-1} + bp^{\lambda-1} + c$, where a, b, c are constants, a and b positive. Since $(1-\eta)^2 + \eta^2 \geq \frac{1}{2}$, we have $\lambda \leq 1$, and hence this denominator is a decreasing function of p.

Thus, assuming $0 < p \leq r$, we find that $p = r$ when $f(p,r)$ is maximal. Let us now consider the case where $0 < r \leq p$. Then $K \leq L$, and the writing masks are submasks of the reading masks. Then (see Kristoferson, 1997) we get equation (24.2) with num $= nMp^{\gamma}r(1-2\beta)^2$ and denom $= 1 + (n-1)(1-2\beta)^2 p^{\kappa-\gamma} + [M-1+(1-2\beta)^2(1-n-M)]p^{\gamma}r + Tp^{1-\gamma}[1 + (n-1)(1-2\beta)^2 p^{\lambda} + (M - 1 + (1-2\beta)^2(n-1)(M-1))pr]$.

Now fix any value of p between 0 and 1. Then $f(p,r)$ is a function of r, where $0 < r \leq p$. If we divide num and denom by r, we get a constant numerator and a denominator of the form $ar^{-1} + b$, where a is a positive constant and b a constant.

Thus, assuming $0 < r \leq p$, we find again that $p = r$ when $f(p,r)$ is maximal. To sum up: Approximately nothing is gained by allowing different probabilities of activation for writing and reading, and so in the rest of this paper we will use just one probability of activation $p = 2^{-K}$. Note that we have considered only Jaeckel's design of SDM and not, for instance, Kanerva's original design. However, the calculations are much more involved for the latter and, given experience with the different designs, it would be surprising if the result were not the same in all cases. Besides, Jaeckel's design, or the sparse version of it discussed below, is what is being used for further work, since it is easier to deal with in many ways and also seems to be biologically the most plausible.

Note the special case where the data are not noisy ($\beta = 0$) but each datum is stored at many noisy addresses. An example might be storing a lot of handwritten letters and then trying to identify another hand-

written letter (the shape of a handwritten letter defining the address, and the name of the letter the datum).

We conclude this section by discussing the assumption that Z is approximately normally distributed. Writing $Z = \sum_t A_t$, where $A_t = \sum_{k=1}^{n} \sum_{h=1}^{M} L_{t,k,h} B_{t,k}$, we cannot claim that the A_ts are independent: An observed relatively large absolute value of a certain A_t indicates that more hard locations than expected have activated $\mathbf{Y}$, giving an increased probability for large absolute values also for the other A_ts. However, the larger the memory, the more the distribution of the number of hard locations activating $\mathbf{Y}$ is concentrated around the expected value Mr, and so the normal approximation should work for large enough memories. There are also arguments using the independence relations concerning the $L_{t,k,h}$s and $B_{t,k}$s, but we cannot go into them here. This kind of somewhat sloppy reasoning, also used in the derivation of equation (24.2) above, has been standard in the literature on SDM. It has been justified by simulations showing agreement with the theory for interesting values of the memory parameters. We plan to test the results of this paper by simulations, too.

24.2 Scaling Up the Memory

What happens to the performance of the memory, i.e. the signal-to-noise ratio ρ, when the memory gets bigger? For instance, what happens to ρ if we keep the "load" $\tau = \frac{T}{M}$ constant and let $M \to \infty$? We will consider only the case where $\eta = \beta = 0$. Putting $r = p$, $\lambda = \beta = 0$, $\kappa = \gamma = -\log_2(1-\epsilon)$, $T = \tau M$ in equation (24.2) and simplifying, we get the following approximate equation,[4] which was already derived in Kristoferson (1995) (assuming $n = 1$):

$$\rho^2 = \frac{Mp^{1+\gamma}}{1 - p^{1+\gamma} + \tau M p^{1-\gamma} + \tau M^2 p^{3-\gamma}} \tag{24.3}$$

where $\gamma = -\log_2(1-\epsilon)$.

The performance is studied in Kristoferson (1995) for different values of M and τ, always using the optimal value of p, i.e. maximizing ρ. It is shown that the optimal p satisfies the following equation:

$$p^{3-\gamma} - \frac{\gamma}{(1-\gamma)M} p^{1-\gamma} - \frac{1+\gamma}{2(1-\gamma)\tau M^2} = 0. \tag{24.4}$$

For $\epsilon = \gamma = 0$ we get the well-known formula

$$p = \sqrt[3]{\frac{1}{2\tau M^2}}. \tag{24.5}$$

[4] From now on we will approximate $M - 1$ by M without further mention.

For $\epsilon > 0$, however, only numerical methods are used in Kristoferson (1995), and no (approximate) explicit formulae for the functions $p(M, \tau)$ and $\rho(M, \tau)$ are given. Here we want to study the asymptotic behavior of p and ρ when $M \to \infty$ (with τ constant, the problem of scaling up, or with τ varying in some sensible way).[5] Then such formulae will be of interest.

To finish the case where $\epsilon = 0$, we get $\rho^2 \approx 1 \Big/ \Big[\tau\Big(1 + 3\sqrt[3]{\frac{1}{4\tau^2 M}}\Big)\Big]$ from equations (24.3) and (24.5). Now $4\tau^2 M \gg 1$ for "normal" values of M and τ, so that

$$\rho \approx \frac{1}{\sqrt{\tau}} \qquad \text{if } \epsilon = 0. \tag{24.6}$$

Furthermore, it is clear that for fixed τ

$$\lim_{M \to \infty} \rho = \frac{1}{\sqrt{\tau}} \qquad \text{if } \epsilon = 0. \tag{24.7}$$

So the memory scales up when $\epsilon = 0$.

This is not the case, however, when $\epsilon > 0$, as first observed by Sjödin (1997). To prove it, consider a given $\epsilon > 0$, M and τ. Observe that

$$\rho_\epsilon = (1 - \epsilon)^{K_\epsilon} \rho_0^* = p_\epsilon^\gamma \rho_0^* \leq p_\epsilon^\gamma \rho_0, \tag{24.8}$$

where:

- p_ϵ and ρ_ϵ are the optimal values of p and ρ for the given $\epsilon > 0$, M, and τ;
- K_ϵ is the corresponding mask-length, i.e. $p_\epsilon = 2^{-K_\epsilon}$;
- ρ_0^* is the value of ρ we get if we change ϵ to 0 without changing $p = p_\epsilon$, and
- ρ_0 is the optimal value for $\epsilon = 0$, M and τ.

Now fix τ and let $M \to \infty$. Then by equation (24.4) $p_\epsilon \to 0$, and by equation (24.7) $\rho_0 \to \frac{1}{\sqrt{\tau}}$. Thus for fixed τ

$$\lim_{M \to \infty} \rho = 0 \qquad \text{if } \epsilon > 0, \tag{24.9}$$

and the memory does not scale up.

Let us now examine in more detail how p_ϵ and ρ_ϵ depend on M and τ if $\epsilon > 0$. Substituting $p_\epsilon = \frac{k}{\sqrt{M}}$ in (24.4) and simplifying, we get

$$\phi(k) \;=\; k^{1-\gamma}\Big(k^2 - \frac{\gamma}{1-\gamma}\Big) - \frac{1+\gamma}{2(1-\gamma)\tau} M^{-\frac{\gamma+1}{2}} \;=\; 0. \tag{24.10}$$

[5] Observe that M shall be $\ll \binom{N}{K} 2^K$, the number of possible hard locations. This implies that N has to grow approximately like $\log M$ when $M \to \infty$.

Observe that $\phi\Big(\sqrt{\frac{\gamma}{1-\gamma}}\Big) < 0$. Furthermore, $\phi\Big(\sqrt{\frac{1}{1-\gamma}}\Big) = \Big(\frac{1}{1-\gamma}\Big)^{\frac{1-\gamma}{2}} - \frac{1+\gamma}{2(1-\gamma)\tau}M^{-\frac{\gamma+1}{2}} > \frac{1}{(1-\gamma)^{\frac{1-\gamma}{2}}}\Big(1 - \frac{M^{-\frac{\gamma+1}{2}}}{\tau(1-\gamma)^{\frac{1+\gamma}{2}}}\Big)$, which is greater than 0 if $\tau^2 M > \frac{\tau^{\frac{2\gamma}{1+\gamma}}}{1-\gamma}$.

For "normal" values of M and τ, $\tau^2 M$ is large enough for this condition to be satisfied, and then we have $\phi(k) = 0$ for some $k = k_\epsilon = k_\epsilon(M,\tau)$ between $\sqrt{\frac{\gamma}{1-\gamma}}$ and $\sqrt{\frac{1}{1-\gamma}}$, i.e.

$$p_\epsilon = \frac{k_\epsilon}{\sqrt{M}} \tag{24.11}$$

where $0 < \sqrt{\frac{\gamma}{1-\gamma}} < k_\epsilon < \sqrt{\frac{1}{1-\gamma}}$. Substituting (24.11) in (24.3) we get, for large enough $\tau^2 M$, $\rho_\epsilon^2 \approx \frac{k_\epsilon^{2\gamma}}{(1+k_\epsilon^2)\tau M^\gamma}$. We now observe that (24.10) implies that $k_\epsilon \to \sqrt{\frac{\gamma}{1-\gamma}}$ when $\tau^2 M \to \infty$, and thus for large enough $\tau^2 M$,

$$\rho_\epsilon \approx \frac{c_\epsilon}{\sqrt{\tau M^\gamma}}, \tag{24.12}$$

where $c_\epsilon = \sqrt{\gamma^\gamma (1-\gamma)^{1-\gamma}}$ lies between $\frac{1}{\sqrt{2}}$ and 1.

This provides another proof of (24.9), and we also find the following generalization of (24.7): if $\tau = \tau_0 M^{-\gamma}$, where τ_0 is a constant, then

$$\lim_{M\to\infty} \rho_\epsilon = \frac{c_\epsilon}{\sqrt{\tau_0}}. \tag{24.13}$$

Thus we keep the performance approximately constant when $M \to \infty$, if we take $T = \tau M$ proportional to $M^{1-\gamma}$. Values of γ around 0.5 or even higher could occur in applications of SDM; hence we would like to do better, i.e. have scaling up also for $\epsilon > 0$.

Dropping the subscript ϵ, we get $\rho \le (1-\epsilon)^K \rho_0$ from (24.8). Recall that $p = 2^{-K}$. Since $p \to 0$ by (24.4), we have $K \to \infty$, and thus $\rho \to 0$, when $M \to \infty$, τ constant. We would like to let K be (a small) constant, but still have optimal values of p. This can be achieved by using something like Jaeckel's hyperplane design (Kanerva, 1993) with sparse addresses, as suggested by Sjödin (1997). Thus let the hard locations be defined by randomly chosen masks with a fixed number K of selected coordinates, and let the addresses be chosen randomly except that for each coordinate the probability of 1 be not $\frac{1}{2}$ as before, but some number α near 0. A hard location is activated by an address if the address has 1s in all selected coordinates. Then $p = \alpha^K$, and we assume that $\alpha = \alpha(M,\tau,\epsilon)$ is chosen so that p be optimized. Here ϵ is the probability

that a bit equal to 1 in the correct address turns into 0 in the reading address (the corresponding probability for a 0 turning into 1 most likely being much lower). Let us see how this works.

We find (Kristoferson, 1995) that the stochastic variable L_t, i.e. the number of hard locations activated by both $\mathbf{Y}$ and $\mathbf{X}_t$, has the distribution $\mathrm{Bin}(M, p^2)$ for $t \geq 1$, and the distribution $\mathrm{Bin}(M, \delta p)$ for $t = 0$, where δ is the constant $(1-\epsilon)^K$. This implies (cf. eq. 24.3)

$$\rho^2 = \frac{M\delta p}{1 - \delta p + \frac{\tau M p}{\delta} + \frac{\tau M^2 p^3}{\delta}}. \tag{24.14}$$

By differentiation it is found that the optimal value of p is (cf. eq. 24.5)

$$p = \sqrt[3]{\frac{\delta}{2\tau M^2}}. \tag{24.15}$$

Equations (24.14) and (24.15) imply $\rho^2 \approx \delta^2 \Big/ \left[\tau\left(1 + 3\sqrt[3]{\frac{\delta^2}{4\tau^2 M}}\right)\right]$. Assuming $4\tau^2 M \gg 1$ we get

$$\rho \approx \frac{\delta}{\sqrt{\tau}} \tag{24.16}$$

and, for fixed τ,

$$\lim_{M\to\infty} \rho = \frac{\delta}{\sqrt{\tau}}. \tag{24.17}$$

So the memory scales up.[6]

We remark that the formulae (24.16) and (24.17) are not very sensitive to the value of p. They hold even if we use, e.g. $p = \sqrt[3]{\frac{1}{2\tau M^2}}$ as in Sjödin (1997).

25 A Fast Activation Mechanism for the Kanerva SDM Memory

Roland Karlsson

The Kanerva sparse distributed memory (SDM; Kanerva, 1988, 1993) is a clever mechanism for storing/retrieving data in/from a memory with a huge address space. The memory addresses, also called input addresses, are N-bit vectors (N is generally large). The actual memory contains substantially fewer memory locations (called *hard locations*) than there are possible input addresses. When accessing the memory, a number of hard locations "close to" the input address are activated.

[6] Since M must now be $\ll \binom{N}{K} \approx \frac{N^K}{K!}$ (the number of possible hard locations), N now has to grow like $M^{\frac{1}{K}}$.

In the basic SDM, each hard location has an N-bit address, and the activation criterion is "close" according to Hamming distance (i.e. the number of mismatched address bits) between the input address and the location's address.

This paper concerns only the activation mechanism for the SDM, but briefly, the write and read operations work as follows: The contents of a hard location is a vector of integers where each integer corresponds to one bit in the input data. When writing to the memory, the data bits (from the input data vector) are added (adding -1 when the data bit is 0 and adding $+1$ when the data bit is 1) to the contents of all activated hard locations. When reading, the content vectors of the activated hard locations are summed and the sum is converted to the output data by some threshold mechanism. The key idea is that data written sufficiently "far away" (i.e. with few common activated hard locations) from the address can be considered noise that will more or less cancel out. Implementing a good threshold mechanism is somewhat involved but details can be found in Sjödin, Karlsson, and Kristoferson (1997).

Several activation mechanisms have been used, e.g. Hamming distance and the Jaeckel selected-coordinate design (Jaeckel 1989a, 1989b). The activation mechanisms have all been based on matching the input address with some "matching pattern" for each hard location. This makes the implementations impractical (except for small problems) without some massively parallel hardware support.

This paper introduces an efficient mechanism for finding the hard locations that are "close to" the input address. The proposed mechanism may be viewed as an efficient implementation of a restricted Jaeckel design. It is suitable for execution on sequential machines or machines with a moderate level of parallelism (say, tens to hundreds of processors). The implementation, apart from being much faster, yields approximately the same signal-to-noise ratio.

The rest of the paper is organized as follows. Section 25.1 describes the activation mechanism in the Jaeckel selected-coordinate design. Section 25.2 describes the activation mechanism in the new proposed design. Section 25.3 compares the performance of the Jaeckel with that of the new design. And finally section 25.4 concludes the paper.

25.1 The Jaeckel Selected-Coordinate Design

In the Jaeckel selected-coordinate design (see fig. 25.1) the activation mechanism is based on selecting an activation pattern for each hard location.

The activation pattern consists of a list of k pairs of bit-number/value. The hard location is said to be "near" the input address if the values

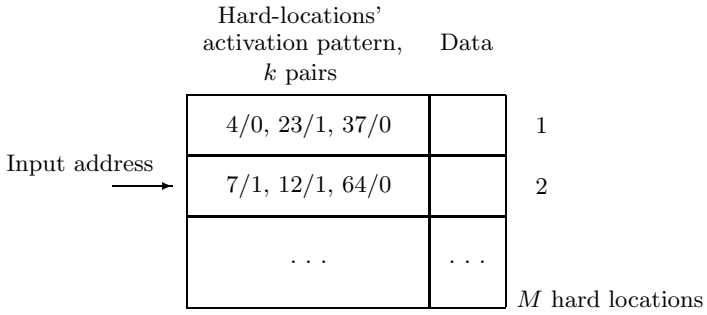

FIGURE 25.1. The Jaeckel Selected-Coordinate Design.

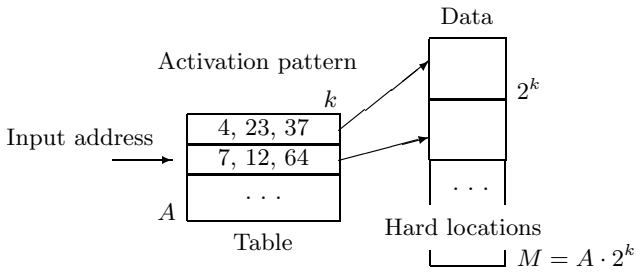

FIGURE 25.2. The new selected-coordinate design.

(of the activation bits) match exactly. In the example in figure 25.1, the first hard location is activated by an input address if the bits 4, 23, 37 in the input address have the values 0, 1, 0.

If the number of activation bits is k, the probability of activation is $p = 2^{-k}$. If the number of hard locations is M, the mean number of activations per access is $\overline{A} = M \cdot 2^{-k}$.

25.2 The New Selected-Coordinate Design

A table is used in the new selected-coordinate design (see fig. 25.2). The number of entries (A) in the table equals the number of desired activations when accessing the memory with an input address. Each entry in the table specifies a subset of k address bits. In the example in figure 25.2 the selected address bits for the first table location are 4, 23, 37. On access, the specified bits (in the input address) are used as an index to a block of 2^k hard locations. In the example this means that 3 bits in the address are used as an index to one of 8 positions in a block. The hard-location memory consists of A such blocks, and therefore the size of the hard-location memory is $M = A \cdot 2^k$.

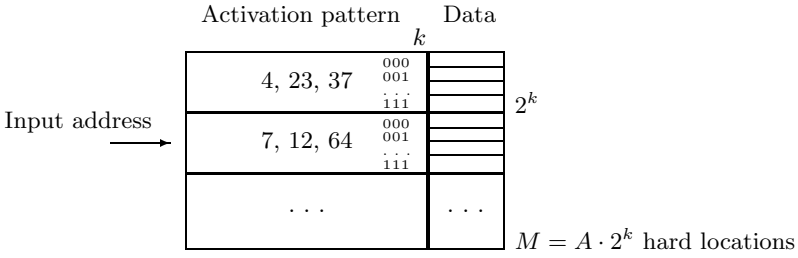

FIGURE 25.3. The new design implemented as a restricted Jaeckel design.

Compared to the earlier designs, the number of access calculations decreases by a factor 2^k. The number of bits (k) is usually in the range of 5–20, making the decrease in access time between 2 and 6 orders of magnitude.

The activation patterns in the table are substantially fewer than in the Jaeckel design (A patterns vs. M patterns). It is not a good idea to generate the activation patterns in the table at random. A bad choice of activation patterns might lead to very unsatisfactory results. To remedy this problem, a restriction is introduced for choosing bits in the activation patterns: A bit cannot be chosen if there exists any other bit that has been used fewer times. Note that this optimization is neither necessary nor used in the Jaeckel model.

The new design can be viewed as a restricted version of Jaeckel's design (see fig. 25.3), with identical (but much slower) functionality since the new design could be implemented by dividing Jaeckel's hard-location memory into A blocks each with 2^k locations, and by using in each block all combinations of 0s and 1s in a fixed subset of the address bits.

The two main advantages of the new design are (1) it is much faster and (2) the number of activations is fixed to A. The latter result is used in Sjödin, Karlsson, and Kristoferson (1997) to get better convergence. One disadvantage may be that it is less random, e.g. there may be problems with interference between the choice of activation pattern and some input data. One constraint on the basic version of the new design is that the number of hard locations is fixed to $M = A \cdot 2^k$. This can of course be remedied by not using all 2^k locations.

25.3 Results

To compare the performance of the new design with Jaeckel's design, we used an artificial benchmark. Random data patterns are written at random addresses. Then we try to retrieve those data by reading at exactly

Table 25.1
Retrieval Error Rates (%) for the Two Designs

Activations (A or $\overline{A}$)	Bits (k)	Address-bit error rate					
		New design			Jaeckel's design		
		0%	1%	2%	0%	1%	2%
8	10	2.6	4.6	6.8	3.6	6.0	8.4
16	9	3.5	5.4	7.6	3.7	5.6	7.7
32	8	4.7	6.5	8.4	5.2	7.0	9.0
64	7	10.3	11.9	13.8	10.0	11.9	13.8
128	6	20.0	21.6	22.7	19.8	21.4	22.9

the same input addresses or some disturbed address. Both designs were also used in a real example: trying to determine which circuit board to replace given a number of error flags.

For the artificial benchmark the tests were carried out for several sizes of hard-location memories and several numbers of stored patterns. The real example used several hard-location memory sizes. The performance of the two designs in all tests was very similar.

To get good statistics for table 25.1, we used an artificial benchmark with a small memory size. Otherwise, the execution times for the Jaeckel design become far too long. The size (M) of the hard-location memory is 8192 positions. The address and data length are both 40 bits. The number of stored patterns is 2048, or 25% of the number of hard locations. For all three runs (0, 1, 2% address-bit error rate) of the benchmark, the results for the new design and for Jaeckel's original design were comparable.

Note though that the Jaeckel results are somewhat worse for 8 activations. When the mean number of activations $\overline{A}$ is small, this is an expected result as some input addresses activate very few (sometimes even zero) hard locations. The new design always activates A hard locations.

25.4 Conclusions

A fast activation method has been found for the SDM memory. Benchmark results show that the new design has a signal-to-noise ratio comparable to that of the Jaeckel selected-coordinate design. The speed-up relative to the Jaeckel (and all older) designs is several orders of magnitude. It can be shown (Kanerva, personal communication) that the method is equivalent to the RAM-based activation of data cells in the WISARD architecture (recognition device) of Wilkie, Stonham, and Aleksander (Aleksander and Morton, 1995).

26 From Words to Understanding

Jussi Karlgren and Magnus Sahlgren

As was discussed in section 22, language is central to a correct understanding of the mind. Compositional analytic models perform well in the domain and subject area they are developed for, but any extension is difficult and the models have incomplete psychological veracity. Here we explore how to compute representations of meaning based on a lower level of abstraction and how to use the models for tasks that require some form of language understanding.

26.1 The Meaning of 'Meaning'

The use of vector-based models of information for the purpose of representing word meanings is an area of research that has gained considerable attention over the last decade. A number of different techniques have been suggested that demonstrate the viability of representing word meanings as semantic vectors, computed from the co-occurrence statistics of words in large text data (e.g. Deerwester et al., 1990; Schütze, 1992; Lund and Burgess, 1996). Unfortunately, the philosophical rationale for this practice has remained tacit, which is remarkable since the vector-based models purport to uncover and represent word meanings. What, one might ask, are those meanings that the words of our language apparently have? Are they perhaps some form of mental concepts that exist in the minds of language users, or are they merely the objects named by the words, and if so, how do we represent something like it in a computer system? It seems that we need to know *what* it is we want to represent before we can start thinking about *how* to represent it in a computer system. Succinctly, it seems as if the recourse to semantics demands an explanation of the meaning of 'meaning'.

Ludwig Wittgenstein suggests in *Philosophical Investigations* (1953) that we should view meaning as something founded in linguistic praxis, and that the meaning of a word is determined by the rules of its use within, as he puts it, a specific *language-game*. This suggestion led to the famous dictum "meaning is use," which is sometimes referred to as a Wittgensteinian theory of meaning. The idea is that to understand the meaning of a word, one has to consider its use in the context of ordinary and concrete language behavior. To know the meaning of a word is simply to be able to use the word in the correct way in a specific language-game or linguistic praxis. This line of reasoning thus allows us to define semantic knowledge as that which we *make use of* when successfully carrying out linguistic tasks. According to this way of thinking, meaning is the vehicle by which language travels.

Thus it is language itself and not the concept of meaning that is primary to Wittgensteinian semanticist. The question about the meaning of 'meaning' must therefore be answered "from within" a theory of language, since words do not (and in a stronger sense *cannot*) have meaning outside language. That is, it does not make any sense to ask what the meaning of a word is in isolation from its use in language, since it is only by virtue of this use that the word has meaning. The lack of rigid designations regarding the concept of meaning facilitates our understanding of language as a dynamic phenomenon. What we need in order to understand the nature of meaning is not so much a rigid definition of the concept of meaning, but rather a profound understanding of the inherent structures of natural language. In short, what we need is a structuralistic account of language.

Using such a relatively agnostic theory of meaning, we will in what follows attempt to exemplify its utility for information-access tasks, arguably the most important and applied of language-technology tasks, and one that relies crucially on some form of textual understanding.

26.2 A Case in Point: Information Access

Text is the primary repository and transmitter of human knowledge. Many other types of knowledge representation have been proposed and used for specific purposes, but for most purposes text has proven efficient, flexible, and compact for generation, storage, and access. But while accessing information in text is simple and unproblematic for a human reader, finding the right text to access may be difficult. Computer systems can be of help here, but to do this, systems must have some form of understanding of text content.

26.2.1 System View of Documents

Information-access systems view documents as carriers of topical information and hold words and terms as reasonable indicators of topic. The techniques used for analysis and organization of document collections are focused primarily on word and term occurrence statistics. Documents and information needs alike are analyzed in terms of words.

Although this simple approach has its obviously effective characteristics, it also has some drawbacks. The results provided by information-access systems of today are unimpressive: by the standard metrics defined and practiced in the field, nothing like optimal performance is delivered by any system. To some extent, this is a problem that has to do with the indeterminacy of the evaluation metrics themselves: Relevance is an ill-defined characteristic of documents. But to a great extent systems do not deliver what should and could be expected of them be-

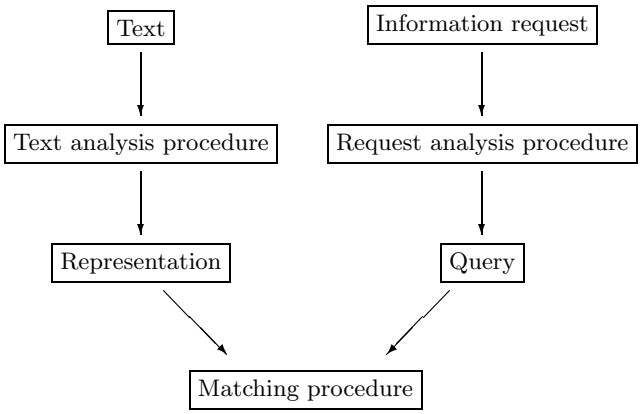

Figure 26.1. The standard model of information retrieval.

cause they model topic and text unsatisfactorily. Although the model is simple and designed not to rely on brittle theory, it does not reflect the underlying structure of textual information transmission sufficiently for purposes of designing useful systems for information access.

26.2.2 The Standard Model for Information Retrieval

The standard model for information retrieval and the basis for most information-system design is roughly as shown in figure 26.1. There is some body of texts; information requests are put to some system that handles this body of texts; the texts are analyzed by some procedure to yield a nontextual representation of them; the information requests are likewise analyzed by an identical or similar procedure to yield a query. The two representations are then matched. The texts with the best matches are presented as potential information sources to fulfill the request.

The point of the analysis is to facilitate matching by (1) reducing the amount of information, to make the representations manageable—it must somehow counter the variability of language and the freedom it affords language users—and (2) resolving the vagueness and indeterminacy inherent in language. The resulting representations are assumed in some way to be alinguistic and amenable to pure formal manipulation.

This quite intuitive and in many ways appealing model hides the complexity of human language use from the matching procedure, which can then be addressed using formal methods. This is not entirely to the benefit of the enterprise. The very same mechanisms that make the matching

complicated—the vagueness and indeterminacy of human language—are what make human language work well as a communicative tool. Awareness of this is typically abstracted out of the search process. The major difference between using an automated information-retrieval system and consulting with a human information analyst is that the latter normally does not require the request to be transformed into some invariant and unambiguous representation; neither does the human analyst require the documents themselves to be analyzed into such a representation. A human analyst not only copes with but utilizes the flexibility of information in human language: It is not an obstacle but an asset. For nontrivial retrieval as performed by humans, concepts glide into each other painlessly and with no damage done to the knowledge representation they utilize, and documents that have previously been thought to be of some specific type or topic can be retrieved for perfectly new and unexpected purposes.

26.3 Words as Content Indicators

The basic drawback of the systems of today is their impoverished picture of text content. They treat texts as containers for words, and words as neat and useful indicators of content. But there is no exact matching from words to concepts. Words are vague, both polysemous and incomplete: every word means several things and every thing can be expressed with any one of several words. Words suffer from the combined drawback of being ambiguous and nonexclusive as indicators of content.

Any representation of content should transcend this inherent ambiguity of words—and this is how our kind of model is intended to improve matters.

26.3.1 The Distributional Hypothesis

Viewed from a structuralistic perspective, natural language may be characterized as a sequence of semantically arbitrary symbols (i.e. words). The symbols are semantically arbitrary since it is not their physical properties that determine their meaning. Rather, it is the relations between the symbols of the sequence that define them. This is to say that the meanings of the symbols in the sequence do not derive from any inherent semantic properties of the symbols.

This relational aspect of meaning can be seen if we consider a polysemous word, for example 'fly', which has a different meaning in the context of A, for example in texts referring to aerial activities, than in the context of B, for example in texts about insects. The reason the word has different meanings in these different contexts is not that it has a different inherent semantic property in the context of A than in the

context of B, for that would mean that a word could change its inherent semantic properties at any time, making linguistic communication, understanding, and hence language itself virtually impossible. The reason for the different meanings of the word is rather that the context of A is different from that of B.

This characterization of natural language as a linear sequence of semantically arbitrary elements allows us to formulate a theory of meaning known as the *distributional hypothesis*. The theory originates from the work of Zellig Harris, who, in his book *Mathematical Structures of Language* (1968, p. 12), states that "the meaning of entities, and the meaning of grammatical relations among them, is related to the restriction on combinations of these entities relative to other entities."

These combinatorial restrictions can be viewed as semantic constraints that govern the distribution of entities in language. That is, if two separate entities occur in combination with the same set of other entities C, the distribution of the two separate entities are governed by the same semantic constraints that are manifested in the distributional pattern that consists in the co-occurrence with C. It is by virtue of this distributional similarity that the entities have similar meaning. This is to say that similarity in distribution implies similar values of semantic information.

The merit of this hypothesis in relation to the Wittgensteinian account of meaning is that the distributional patterns of words can be thought of as manifesting language use. That is, the use of a word is manifested by its distribution in language, which in turn is defined by the contexts in which the word occurs. This means that the context could be utilized as a measure of distribution, by which the use (and thus also the meaning) of a word could be determined. Thus, if the *meaning* of a word is determined by its *use* in language, and its use is manifested by its *distribution*, the distributional patterns as defined by the *contexts* of a word can be seen as viable tools for determining the meaning of that word.

The idea is that words are semantically similar to the extent that they share contexts. If two words w_1 and w_2, say 'beer' and 'wine', frequently occur in the same context C, say after 'drink', the hypothesis states that w_1 and w_2 are semantically related, or, stated more strongly, that they are semantically similar. The semantic similarity (or relatedness) of 'beer' and 'wine' is thus due to the similarity of usage of these words. This means that the categorization of both words as, for example, referring to alcoholic beverages is possible only because we use them in such a way; for example, after the word 'drink' and in the vicinity of the word 'drunk'. The categorization is not a *cause* of usage but a *consequence*.

In an attempt to formalize these ideas, we could say that the meaning

M of a word w is determined by its distribution D in text T. D over T can be defined as the union of the contexts C in which w occurs. Thus, if $D(w, T)$ (the distribution of w in T) determines $M(w)$ (the meaning of w), and $D(w, T)$ equals $\sum\{C \mid w \in C\}$ (the contexts in which w occurs), this can be seen as a representation of $M(w)$. This representational scheme thus justifies the claim that word meanings *can* be uncovered and represented in computer systems. It also justifies the claim that this can be done without forcing us to commit to any particular ontology about *what* these meanings are. Rather, it is because we do not demand any rigid definition of the concept of meaning that this representational scheme becomes feasible.

26.4 Latent Semantic Analysis

This representational scheme is the communal rationale for vector-based semantic analysis. The assumption that "words with similar meanings will occur with similar neighbors if enough text material is available" (Schütze and Pedersen, 1997) is central to all the various approaches. What separates them is *how* they implement the idea. The pioneering technique in this area of research, latent semantic analysis (LSA) (Landauer and Dumais, 1997), collects the text data in a words-by-documents co-occurrence matrix where each cell indicates the frequency of a given word in a given text sample of approximately 150 words. The words-by-documents matrix is normalized by using logarithms of word frequencies and entropies of words across all documents. The normalized matrix is then transformed with singular-value decomposition into a much smaller matrix. This dimension reduction appears also to accomplish inductive effects, reminiscent of human psychology, by capturing latent semantic structures in the text data. Words (or, more to the point, *concepts*) are thus represented in the reduced matrix by semantic vectors of dimensionality n (300 proving to be optimal in Landauer and Dumais's 1997 experiments).

A similar approach is taken by Schütze and Pedersen (1997), who represent the text data as a words-by-words co-occurrence matrix where each cell records the number of times that the word pair occurs in a window spanning 40 words. However, such a matrix with $v^2/2$ distinct entries, where v is the size of the vocabulary, becomes computationally intractable for large vocabularies, so they first approximate the matrix using class-based generalization in two steps and then transform it with singular-value decomposition, so that words are represented in the final reduced matrix by dense semantic vectors of dimensionality n ($n = 20$ in Schütze and Pedersen, 1997). The motivation for reducing the dimensionality of the approximated matrix with singular-value decomposition

is, as in LSA, that it improves generalization and makes the representations more compact.

This kind of representational scheme where words are represented as semantic vectors that are calculated from the co-occurrence statistics of words in large text data has proven to be both computationally advantageous and cognitively justified. The drawback of singular-value decomposition is that it places heavy demands on computing time and memory, which suggests that an alternative way of achieving the inductive effects of dimension reduction might be worth considering. A number of techniques for doing so have been proposed under such names as random mapping (Kaski, 1998), random projections (Papadimitriou et al., 1998), and random indexing (Kanerva et al., 2000), and they have the same underlying mathematics.

26.5 Random Indexing

In previously reported experiments with random indexing (Kanerva et al., 2000), documents of approximately 150 words each are represented as high-dimensional random index vectors (dimensionality $> 1{,}000$) that are accumulated into a words-by-documents matrix by adding a document's index vector to the row for a given word every time the word appears in that document. The method is comparable to LSA, except that the resulting matrix is significantly smaller than the words-by-documents matrix of LSA, since the dimensionality of the index vectors is smaller than the number of documents. By comparison, assuming a vocabulary of 60,000 words in 30,000 documents, LSA would represent the data in a $60{,}000 \times 30{,}000$ words-by-documents matrix, whereas the matrix in random indexing would be $60{,}000 \times 1{,}800$ when 1,800-dimensional index vectors are used. This seems to accomplish the same inductive effects as those attained by applying singular-value decomposition to the much larger matrix, but without the heavy computational load that singular-value decomposition requires.

In the present experiment, the high-dimensional random vectors of random indexing have been used to index *words* and to accumulate a words-by-contexts matrix by means of *narrow* context windows consisting of only a few adjacent words on each side of the focus word. As an example, imagine that the number of adjacent words in the context window is set to two. This would imply a window size of five space-separated linguistic units, i.e. the focus word and the two words preceding and succeeding it. Thus the context for the word 'is' in the sentence 'This parrot is no more' is 'This parrot' and 'no more', as denoted by

[(This parrot) is (no more)].

Calculating semantic vectors using random indexing of words in narrow context windows is done in two steps. First, an n-dimensional sparse random vector called a *random label* is assigned to each word type in the text data. These labels have a small number k of randomly distributed -1s and $+1$s, with the rest set to 0. The present experiment utilized 1,800-dimensional labels with $k = 8.7$ on average, with a standard deviation of ± 2.9. Thus a label might have, for example, four -1s and six $+1$s. Next, every time a given word—the focus word f_n—occurs in the text data, the labels for the words in its context window are added to its *context vector*. For example, assuming a $2 + 2$ sized context window as represented by:

$$[(w_{n-2}w_{n-1})f_n(w_{n+1}w_{n+2})]$$

the context vector of f_n would be updated with:

$$L(w_{n-2}) + L(w_{n-1}) + L(w_{n+1}) + L(w_{n+2})$$

where $L(x)$ is the label of x. This summation has also been weighted to reflect the distance of the words to the focus word. The weights were distributed so that the words immediately preceding and succeeding the focus word get more significance in the computation of the context vectors. For the four different window sizes used in these experiments, the window slots were given weights as follows:

1 + 1: [(1) 0 (1)]
2 + 2: [(0.5, 1) 0 (1, 0.5)]
3 + 3: [(0.25, 0.5, 1) 0 (1, 0.5, 0.25)]
4 + 4: [(0.1, 0.1, 0.1, 1) 0 (1, 0.1, 0.1, 0.1)].

The rationale for these operations is that a high-dimensional context vector, by effectively being the sum of a word's local contexts, represents the word's relative meaning. This means that we will be able to model word content with some confidence. But texts are more than words and their content.

26.6 What Is Text, from the Perspective of Linguistics?

The model described above covers much of what we want in terms of human linguistic behavior. But what we know about language and text certainly motivates a more sophisticated model than set theory on the level of word occurrences. Linguists treat linguistic expressions as being composed of words that form clauses that in turn form text or discourse. Words have predictable situation-, speaker-, and topic-*independent* structure that is described formally. Clauses have largely predictable situation-, speaker-, and topic-*independent* structure that is described formally. This is how far formal linguistic analysis takes

us. Attempts at formal analysis at the next level—of text and topic structure—have been only partially successful. Texts have largely unpredictable situation-, speaker-, and topic-*dependent* structure, which cannot be handled adequately with the theoretical apparatus available to us today. Clause structure is connected only indirectly to topicality: mostly it accounts for the local organization of the clause. However, the invariant and predictable nature of clause structure certainly encourages further attempts at building theories that relate meaning to clause structure, and it would be foolish to build a text model that can take no account of recent advances in formal analysis of text structure.

26.6.1 Beyond Word Co-occurrence—Implementing Linguistics

The only property of text that is being utilized in the creation of the context vectors is the distributional patterns of linguistic entities. This comprises, however, only a small fraction of the structural complexity of large texts. There are other inherent structural relations in natural language that might be significant for uncovering semantic information. The distributional hypothesis does tell a story about the foundation of meaning, but it might not tell the *whole* story. If the overall goal of the research is to understand how meaning resides in language, and how to implement linguistic knowledge about meaning in computers, it seems unmotivated not to take these more complex linguistic features into account. Therefore, we have evaluated the method using different degrees of linguistic preprocessing of the training data, such as morphological analysis and part-of-speech tagging, with the intention of investigating whether the utilization of more sophisticated linguistic information in some way concretizes the semantic information captured in the context vectors. To ensure state-of-the art performance in linguistic analysis, we used the functional dependency grammar of English—the FDG parser—developed by Conexor to analyze the text and its words (Järvinen and Tapanainen, 1997).

26.7 The TOEFL-Test

To repeat the fundamental hypothesis of this investigation, the *raison d'être* of the high-dimensional context vectors described in above sections is that they represent the relative meaning of words, and that they therefore can be used to calculate semantic similarities between words.

We will verify this hypothesis by letting the system perform a synonym test. One such test is TOEFL (test of English as a foreign language), which is a standardized test employed, for example, by American universities to evaluate foreign applicants' knowledge of the English lan-

guage. In the synonym-finding part of the test, the test taker is asked to find the synonyms to certain words. For each given word, a choice from four alternatives is provided, where one is the intended synonym and is supposed to be indicated by the person taking the test. In the present experiment, 80 test items of this type were used.

When performing the synonym test, the system simply calculates the distances (the cosine of the angle between context vectors) between the target word and the four alternatives and gives the highest-ranking alternative as its answer (i.e. the one that correlates most closely with the target word).

26.8 Experimental Set-Up

The text data used as learning material for the system was a ten-million-word corpus of unmarked English with a vocabulary of 94,000 words. In the first stage of preprocessing the number of unique word types was reduced based on frequency, by weighting the least and the most frequent words with 0 when they appeared in the context window. A frequency range of 3–14,000 was used and resulted in a vocabulary of 51,000 words. Next, a rather crude method for morphological analysis was implemented by truncating the words. The idea was to approximate word stems by simply chopping off the words at a certain predefined number of letters. In the present experiment, truncation lengths of 6, 8, 10, and 12 were used. As a comparison to the crude truncation approach to morphology, the the Conexor FDG parser was used to analyze the text initially and extract the base form of each word.

The Conexor FDG parser was also used to supply the analyzed text (the text consisting of proper word stems) with part-of-speech information in an attempt to deal with the ever-present ambiguity of languages, the problem being that the same "word" can have several meanings, and many of these orthographically identical but semantically dissimilar words belong to different parts of speech. For example, *roll* can be used as a verb or as a noun. Providing part-of-speech information by simply adding the part-of-speech tag to the beginning of each word would enable the system to detect this kind of ambiguity and to discriminate between these words. For example, the verb *roll* would become *vroll*, whereas the noun *roll* would become *nroll*.

26.9 Results and Analysis

The results of the TOEFL-test are summarized in table 26.1. The numbers in the cells are averages over five runs. The standard deviation for these results is ± 1.5. All results are given in percent of correct answers to the TOEFL-test. The numbers in boldface are the results from ex-

Table 26.1
Average Results (±1.5) in Percent of Correct Answers to the TOEFL-test
Tr. means truncation length, **WS** means 'word stems', and **PoS+WS** means 'part-of-speech tagged word stems'.

Linguistic analysis	Context window				Average
	1 + 1	2 + 2	3 + 3	4 + 4	(±0.73)
None	64.5	67.0	65.3	65.5	65.6
Tr. 6	55.0	57.5	57.3	55.3	56.3
Tr. 8	61.5	64.3	62.0	63.3	62.8
Tr. 10	66.0	68.5	66.3	66.3	66.8
Tr. 12	64.8	65.3	63.8	64.8	64.6
WS	**63.5**	**70.8**	**72.0**	**66.0**	**68.1**
PoS+WS	**66.0**	**64.5**	**65.0**	**65.5**	**65.3**
Average (±0.56)	63.0	65.4	64.5	63.8	

periments with linguistically analyzed text. By comparison, tests with LSA on the same text data, using the LSIBIN program from Telecordia Technologies, produced top scores at 600 factors of 58.75% using the unnormalized words-by-documents matrix, and 65% using a normalized one. The average result reported by Landauer and Dumais (1997) with LSA (using normalization and different text data) is 64.4%, while foreign (non-English-speaking) applicants to U.S. colleges average 64.5%.

These results indicate that high-dimensional random labeling of words in narrow context windows captures similarity relations between words just as effectively as singular-value decomposition of the words-by-documents matrix does (e.g. LSA), as measured by a standardized synonym test. Without using linguistic information, the system averages 65.6% over the four different window sizes used in these experiments. However, already when utilizing a rather naive kind of morphological analysis in the form of carefully applied truncation (using a truncation length of 10 characters), the system's average result increases to 66.8% correct, although it seems imperative not to truncate too early, since this gravely affects the results. Shortening the truncation length to eight characters decreases the result to 62.8%, and shortening it to six renders a meager 56.3%. Extending the truncation length to twelve characters also decreases the result, with a 64.6% average.

The best results were produced by proper stemming of words, which yields 68.1% correct on the average. This indicates that since the inclusion of morphology in the form of proper word-stem analysis or carefully applied truncation yields the best overall results, taking advantage of other inherent structural relations in text, in addition to the distri-

butional patterns of linguistic entities, really might be significant for uncovering semantic information from text data. However, adding part-of-speech information did not further improve the performance. The average result when adding part-of-speech information to the morphologically analyzed text drops to 65.3%. This could be a result of the increase in the size of the vocabulary, which is the consequence of supplying part-of-speech information for each word.

Turning now to the different window sizes, the table shows that a minimal context window with just one word on each side of the focus word yields the worst average result. This might not be surprising, since it seems reasonable to assume *a priori* that a minimal context window will not provide enough contextual information for making the comparison of distributional similarity reliable. This assumption is not categorically supported by the results, however, since for the part-of-speech-tagged word stems, a $1 + 1$ sized context window actually produces the best average result. The results peak in the range of two to three words on each side of the focus word, with a $2 + 2$ sized context window producing the best average result of 65.4%, but with a $3 + 3$ sized context window producing the best individual result of 72% using the morphologically analyzed text. A $4 + 4$ sized context window is only slightly better (63.8%) than the minimal context window, and our experiments with context windows exceeding four words on each side of the focus word gave much lower scores.

26.10 Some Cognitive Implications

The results from these experiments demonstrate that the technique is capable of achieving comparatively good results on a standardized synonym test. The test is designed to measure word knowledge, which would indicate that any subject capable of performing the test with scores above the level of guessing (which statistically would yield 25% correct) possesses a certain amount of linguistic knowledge about word meanings. Therefore, the test could also be seen as a rudimentary intelligence test. Landauer et al. (1998) point out that "word-word meaning similarities are a good test of knowledge—indeed, vocabulary tests are the best single measure of human intelligence."

The results achieved in these investigations are approximately parallel to the results accomplished by foreign applicants to American universities. The question is, then, if the results, viewed from this perspective, justify the conclusion that the system has acquired and applied linguistic knowledge (about word meanings)? Do the high-dimensional random distributed representations constitute a viable model of semantic knowledge? In short: What are the cognitive implications of the accomplish-

ments of the system?

The keyword in this discussion is *functionality*. The performance of the system could be described as *functionally* equivalent to the linguistic behavior of a human language user in carrying out the specific predefined linguistic task of picking out synonyms of a target word. This means that since the system's internal representations in the form of context vectors have been proven (by the successful execution of the TOEFL-test) to be functional for purposes pertaining to linguistic competence, we may describe the system as having acquired and applied the *computational equivalent* of the linguistic knowledge that humans possess when discriminating between word meanings.

This characterization of the system means that the relevant question is *not* whether the system's internal representations of word meanings actually *mean* anything (i.e. if they somehow correspond to how word meanings are represented in the human mind—assuming this question is meaningful), but rather whether they can be *utilized* for the purpose of modeling observable linguistic behavior. The semantic information that the context vectors carry does not reside in the vector representations alone, but rather in the relations between the vectors. The representation is relative rather than absolute, since it is only in relation to each other that the context vectors *mean* anything. The important point is therefore that the system's internal representations can produce linguistic behavioral patterns that manifest semantic knowledge—and that can be regarded a fragment of the *functionality* of a language user. In other words: It is by virtue of letting the meaning of 'meaning' remain indeterminate that we may consider the implementation of a functional pattern as an epistemic or cognitive achievement.

26.11 Implications for Information Access

More concretely, for immediate attention, if we wish to improve on information-access systems—if we use the standard architecture as delineated in the beginning of this chapter—there are three access points, points where the character of the internal text representation influences the working of the system as a whole, and points where a more adaptive and humanlike processing would make a difference:

1. the intelligent selection of document descriptors for each document;
2. the flexible internal expression of those items; and
3. the negotiable elicitation of information needs from the reader.

And this functionality should not be restricted to a single language.

26.12 Meaning in Text

The reason for using narrow context windows to calculate semantic word vectors as opposed to using whole documents, as in LSA, is the assumption that the semantically most significant context is the immediate vicinity of a word. That is, one would expect the words closest to the focus word to be more important than the words further away in the text for determining the meaning of the focus word. The intuition is that a local context is more reliable for measuring semantic similarity between words than a large context region spanning hundreds of words.

This intuition is expressed, for example, by Lin (1997), who states that "two different words are likely to have similar meanings if they occur in identical local contexts." Schütze and Pedersen (1997) argue that local co-occurrence statistics are both qualitatively and quantitatively more informative than document-based co-occurrence statistics, since the number of co-occurrence events will be higher when using a sliding window to define the co-occurrence region than when using documents, especially if the documents are long. Burgess and Lund (2000) also report on the merits of narrow context windows.

If the assumption is correct that local co-occurrence statistics give a more reliable measure of distributional similarity between words than do document-based co-occurrences, one should be able to discern an increase in performance when using narrow context windows, as opposed to documents, for calculating the semantic word vectors. The results of our experiments seem to favor this assumption. Compared to the performance of techniques based on context regions in excess of a hundred words, such as LSA, narrow context windows perform well in at least one linguistic task (TOEFL) pertaining to lexical semantic knowledge.

This improved performance raises the question of whether there might be a difference in what sort of semantic information can be extracted by considering different amounts of context. A larger context might give better clues to what a particular word is about than to what it means. The idea is that two words that are about similar things will occur in similar context regions (e.g. documents), while two words that have similar meanings will occur with similar context neighbors (i.e. words). This means that larger contexts might be more suited for tasks pertaining to topical information, such as information retrieval, than in tasks directed specifically toward lexical semantic competence. The applicability of LSA to information retrieval is well documented (e.g. Dumais et al., 1988; Deerwester et al., 1990), supporting this assumption.

The possibility of a discrepancy between the kinds of semantic information carried by different context sizes suggests that although a syn-

onym test is a fairly reliable method for measuring one kind of semantic knowledge, other conceivable methods for measuring semantic knowledge might be worth considering. Other evaluation procedures have been reported in the literature, such as comparing vector similarities with reaction times from lexical priming studies (Lund and Burgess, 1996) or using LSA for evaluating the quality of content of student essays on given topics (Landauer et al., 1997).

Meaning, the main object of our study, is most decidedly situation-dependent. While much of meaning appears to achieve consistency across usage situations, most everything *can* be negotiated on the go. Human processing appears to be flexible and oriented toward learning from prototypes rather than learning by definition: Learning new words and adding new meanings or shades of meaning to an already known word do not require a formal retraining process. And, in fact, natural use of human languages does not make use of definitions or semantic delimitations; finding an explicit definition in natural discourse is a symptom of communicative malfunction, not of laudable explicitness.

A text model should model language *use* rather than language in the abstract. We need a better understanding of how meaning is negotiated in human language usage: Fixed representations do not seem practical and do not reflect observed human language usage. We need a more exact study of inexact expression, of the *homeosemy* ('homeo' from Greek *homoios* similar) or near and close synonymy of expressions of human language. This means we need to understand the temporality, saliency, and topicality of terms, relations, and grammatical elements—it means modeling the life cycle of terms in language, the life cycle of referents in discourse, and the connection between the two. These experiments have taken but some first steps in that direction.

References

Aleksander, I. and Morton, H. (1995). *An Introduction to Neural Computing*, second edition. London: International Thomson Computer Press.

Boden, M. B. and Niklasson, L. F. (1995). Features of distributed representation for tree structures: A study of RAAM. In L. F. Niklasson and M. B. Boden (eds.), *Current Trends in Connectionism.* Hillsdale, NJ: Erlbaum.

Burgess, C. and Lund, K. (2000). The dynamics of meaning in memory. In E. Dietrich and A. B. Markman (eds.), *Cognitive dynamics: Conceptual change in humans and machines.* Mahwah, NJ: Lawrence Erlbaum Associates.

Chalmers, D. J. (1990). Syntactic transformations on distributed representations. *Connection Science* 2(1–2):53–62.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the Society*

for Information Science 41(6):391–407.

Dumais, S. T., Furnas, G. W., Landauer, T. K., and Deerwester, S. (1988). Using Latent Semantic Analysis to improve information retrieval. *Proceedings of CHI'88: Conference on Human Factors in Computing* (pp. 281–285). New York: ACM.

Durrett, R. (1995). *Probability: Theory and Examples.* 2nd ed. Belmont, Calif.: Duxbury Press, Wadsworth Publishing Co.

Eliasmith, C, and Thagard, P. (2001). Integrating structure and meaning: A distributed model of analogical mapping. *Cognitive Science* 25(2):245–286.

Fodor, J. A. and Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition* 28:3–71.

Gayler, R. W. and Wales, R. (1998). Connections, binding, unification, and analogical promiscuity. In K. Holyoak, D. Gentner, and B. Kokinov (eds.), *Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences* (Proc. Analogy '98 workshop, Sofia), pp. 181–190. Sofia: New Bulgarian University.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science* 7(2):155–170.

Gentner, D., Holyoak, K. J., and Kokinov, B. K. (eds.) (2001). *The Analogical Mind: Perspectives from Cognitive Science.* Cambridge, MA: MIT Press.

Gentner, D. and Markman, A. B. (1993). Analogy—watershed or Waterloo? Structural alignment and the development of connectionist models of analogy. In C. L. Giles, S. J. Hanson, and J. D. Gowan (eds.), *Advances in Neural Information Processing Systems 5* (NIPS '92, pp. 855–863). San Mateo, CA: Morgan Kaufmann.

Harris, Z. (1968). *Mathematical Structures of Language.* New York: Interscience Publishers.

Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence* 46(1–2):47–75.

Hofstadter, D. R. (1984). *The Copycat Project: An Experiment in Nondeterminism and Creative Analogies.* AI Memo 755, Artificial Intelligence laboratory, Massachusetts Institute of Technology.

Hofstadter, D. R. (1985). *Metamagical Themas: Questions of the Essence of Mind and Pattern.* New York: Basic Books.

Jaeckel, L. A. (1989a). *An alternative design for a Sparse Distributed Memory.* Report RIACS TR-89.28. Research Institute for Advanced Computer Science, NASA Ames Research Center.

Jaeckel, L. A. (1989b). *A class of designs for a Sparse Distributed Memory.* Report RIACS TR-89.30, Research Institute for Advanced Computer Science, NASA Ames Research Center.

Järvinen, T. and Tapanainen, P. (1997). *Functional Dependency Grammar.* Publications of the Department for General Linguistics, University of Helsinki.

Kanerva, P. (1988). *Sparse Distributed Memory.* Cambridge, Mass.: MIT Press.

Kanerva, P. (1992). Associative-memory models of the cerebellum. In I. Aleksander and J. Taylor (eds.), *Artificial Neural Networks, 2* (Proc. ICANN '92, Brighton, UK, pp. 23–34). Amsterdam: Elsevier.

Kanerva, P. (1993). Sparse Distributed Memory and related models. In M. H. Hassoun (ed.), *Associative Neural Memories*. New York: Oxford University Press.

Kanerva, P. (1996). Binary spatter-coding of ordered K-tuples. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff (eds.), *Artificial Neural Networks* (Proc. ICANN'96, Bochum, Germany, pp. 869–873). Berlin: Springer.

Kanerva, P., Kristofersson, J., and Holst, A. (2000). Random Indexing of text samples for Latent Semantic Analysis. In L. R. Gleitman and A. K. Josh (eds.), *Proceedings of the 22nd Annual Conference of the Cognitive Science Society,* p. 1036. Mahwah, New Jersey: Erlbaum.

Kaski, S. (1998). Dimensionality reduction by random mapping: Fast similarity computation for clustering. *Proceedings of the IJCNN'98, International Joint Conference on Neural Networks* (Anchorage, vol. 1, pp. 413–418). Piscataway, NJ: IEEE Press.

Kristoferson, J. (1995). *Best probability of activation and performance comparisons for several designs of Sparse Distributed Memory*. Report SICS R95:09, Swedish Institute of Computer Science.

Kristoferson, J. (1997). *Some results on activation and scaling of Sparse Distributed Memory*. Report SICS R97:04, Swedish Institute of Computer Science.

Landauer, T. K. and Dumais, S. T. (1997). A solution to Plato's problem: The Latent Semantic Analysis theory of acquisition, induction and representation of knowledge. *Psychological Review* 104(2):211–240.

Landauer, T. K., Laham, D., Rehder, B., and Schreiner, M. E. (1997). How well can passage meaning be derived without using word order? A comparison of Latent Semantic Analysis and humans. In M. G. Shafto and P. Langley (eds.), *Proceedings of the 19th annual meeting of the Cognitive Science Society,* pp. 412–417. Mawhwah, NJ: Erlbaum.

Landauer, T. K., Laham, D., and Foltz, P. W. (1998). Learning human-like knowledge by Singular Value Decomposition: A progress report. In M. I. Jordan, M. J. Kearns and S. A. Solla (eds.), *Advances in Neural Information Processing Systems* 10, pp. 45–51. Cambridge, Mass.: MIT Press.

Lin, D. (1997). Using syntactic dependency as local context to resolve word sense ambiguity. *Proceedings of ACL-97*, Madrid, Spain.

Lund, K. and Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments & Computers* 28(2):203–208.

Mitchell, M. (1993). *Analogy-Making as Perception: A Computer Model*. Cambridge, MA: MIT Press.

Papadimitriou, C. H., Raghavan, P., Tamaki, H., and Vempala, S. (1998). Latent Semantic Indexing: A probabilistic analysis. *Proceedings of the 17th*

ACM Symposium on the Principles of Database Systems, pp. 159–168. ACM Press.

Plate, T. A. (1994). *Distributed Representation and Nested Compositional Structure*. Ph.D. Thesis. Graduate Department of Computer Science, University of Toronto.

Pollack, J. P. (1990). Recursive distributed representations. *Artificial Intelligence* 46(1–2):77–105.

Rachkovskij, D.A. and Kussul, E.M. (2001). Binding and normalization of binary sparse distributed representations by Context-Dependent Thinning. *Neural Computation* 13(2):411–452.

Rumelhart, D. E, and McClelland, J. L. (1986). On learning the past tenses of English verbs. In J. L. McClelland and D. E. Rumelhart (eds.), *Parallel Distributed Processing 2: Applications* (pp. 216–271). Cambridge, Mass.: MIT Press.

Schütze, H. (1992). Dimensions of meaning. *Proceedings of Supercomputing* (Minneapolis. pp. 787–796). Los Alamitos, CA: IEEE Computer Society Press.

Schütze, H. and Pedersen, J. O. (1997). A cooccurrence-based thesaurus and two applications to information retrieval. *Information Processing and Management* 33(3):307–318.

Sjödin, G. (1995). *Improving the capacity of SDM.* Report R95:12, Swedish Institute of Computer Science.

Sjödin, G. (1996). Getting more information out of SDM. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff (eds.), *Artificial Neural Networks—Proc. ICANN '96,* 477–482. Berlin: Springer.

Sjödin, G. (1997). The Sparchunk code: A method to build higher-level structures in a sparsely encoded SDM. *Proc. 1998 IEEE International Joint Conference on Neural Networks* (IJCNN/WCCI, Anchorage, Alaska, May 1998, vol. 2, pp. 1410–1415). Pincataway, NJ: IEEE Press. Reprinted in this volume.

Sjödin, G., Karlsson, R., and Kristoferson, J. (1997). Algorithms for efficient SDM. *Proc. 1997 Real World Computing Symposium* (RWC'97, Tokyo, January 1997), 215–222. Report RWC TR-96001, Real World Computing Partnership, Tsukuba Research Center, Japan.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence* 46(1–2):159–216.

Wittgenstein, L. (1953). *Philosophical Investigations.* Translated by G. E. M. Anscombe. Oxford: Blackwell. Swedish translation *Filosofiska undersökningar* by A. Wedberg (1992). Stockholm: Thales.

Index

(c, η)-realizable, 222

absolute loss, 213
additional average logarithmic loss, 197
additive decomposition, 157–159, 171, 175
analogical inference, 266, 269
analogical mapping, 269
analogical retrieval, 265
analogy, 254–272
annealing schedule, 154, 156, 159–161, 165, 170, 185
average fitness, 126, 128–130, 134, 135, 137–139, 141, 152, 160–161, 177

bandit problem, 228
Bayes Dirichlet, 172, 173
BayesBuilder, 93, 97
Bayesian information criterion, 173, 175
Bayesian network, 21, 29, 74, 166, 171–175
Bayesian network based on neural networks, 30
Bayesian prior, 165–170
BAYONET, 32
BDe, *see* Bayes Dirichlet
Beatles data, 102, 111, 114, 116, 118
BEDA, *see* Boltzmann estimated distribution algorithm
behavioral psychology, 227
belief network, 29
better-than-worst property, 218
BIC, *see* Bayesian information criterion
Boltzmann distribution, 154–165, 170, 175, 178
Boltzmann estimated distribution algorithm, 155, 159, 162
Boltzmann Machine, 74, 79
  learning, 79, 81, 85
Boltzmann-Gibbs distribution, 75, 79, 85

chunking, 263, 272, 273, 277
circuit
  computer, 253, 257
  neural, 253
clean-up, *see* item memory
clique, 77, 88, 89
complexity, 77
computer analogy of the brain, 256
conditional probability, 142
conditioned mutual information, 39
constraint optimization, 170–171
context vector, *see* semantic vector
context window, 299–301, 304, 305, 307
context-dependent thinning, 254

Copycat, 266
covariance, 141, 143
critical population size, 163
crossover, *see* recombination

deceptive, 132, 151, 168
dense coding, 254, 272, 273, 276, 279, 299
diagnostic decision support system, 94
dialogue-based learning, 27, 55
directed acyclic graphical model, 74
distributed cooperative Bayesian learning strategy
  of type 1, 197
  of type 2, 201
distributed learning system, 190
distributed representation, 253, 254, 257, 261, 269, 270, 272, 305
distributional hypothesis, 297–299, 302
distributional similarity, 298, 305, 307
distributivity, 262, 263

emulation, *see* universal emulator
entropy, 39
explaining away, 91
extended functional connectivity analysis, 53

F-Wright, *see* Fast Wright
factorization, 157–159, 162, 165, 170, 171, 174–176, 178, 185
factorized distribution algorithm, 148, 152, 157–159, 163, 167, 168, 170, 171, 174, 184, 185
Fast Wright, 180, 183
fast-activation SDM, 289–293
FDA, *see* factorized distribution algorithm
fitness landscape, 136, 137, 150, 184
functions
  Int, 164
  OneMax, 145, 163, 167, 174, 180
  Royal Road, 147
  Saw, 150

generalization, 257, 267–269, 299
genetic algorithm, 26, *see* simple genetic algorithm
graphical models, 20

hellinger loss, 213
heritability, 140, 141, 143, 150, 152
hierarchical model, 191
higher-level representation, 261, 272–273, 276, 279
holistic mapping, 264–266, 270
holographic reduced representation, 254, 269, 271–273
HRR, *see* holographic reduced representation
hyperplane design SDM, 273, 276, 279, 288

independent component analysis, 49
information access, 295–297, 306
information content of memory, 273
Int, *see* functions
internal medicine, 95
intractable, 73–75, 78, 80, 87
item memory, 254, 262, 264, 273, 276, 278, 279, 283

Jensen's inequality, 80, 88

Kalman filter, 101–104
  prediction, 115
  switching, 105, 106, 110, 115
  training, 112, 113
Kalman smoothing, 104, 116, 117
kernel mapping, 268–271

language
  ambiguity as asset, 272, 297
  ambiguity of, 303
  flexibility of, 297
  vagueness and indeterminacy of, 297
latent semantic analysis, 299–300, 304, 307
learning, 80, 81, 83
learning factorized distribution algorithm, 172, 174–176
learning from examples, 27, 55, 255, 266–269, 271, 308
Legendre transformation, 80, 82
LFDA, *see* learning factorized distribution algorithm
linear response theory, 75, 83
linkage equilibrium, 125, 126, 128, 129, 133, 143
lob-pass game, 227
lob-pass playing machine, 231
Lob-Pass Problem, 226, 231
local field, 90
LSA, *see* latent semantic analysis

marginal distribution, 126, 127, 129, 130, 133, 145, 168, 184
matching shoulders, 228
MDL, *see* minimal description length
ME-loss, 215
mean field
  approximation, 84
  equations, 76, 83
  free energy, 82
  theory, 73, 74, 76, 80
meaning, literal and figurative, 255, 257, 265
medical diagnosis, 73, 77, 93, 94
minimal description length, 172, 173
mixture of Elman networks, 25
mixture of Gaussians, 24
mixture of predictors, 24
moment, 144, 160, 161
multiple attribute learning, 28
multivariate information analysis, 23, 38
music transcription, 73, 100
mutual information, 39

neural network model, 25
nondistributed Bayesian learning strategy, 195

OneMax, *see* functions

partition function, 81
patterns as symbols, 269
plain model, 191
pointers for representing structure, 253
population learning, 191
potential, 86, 87, 89
prior, *see* Bayesian prior
probabilistic constraint program, 23
probing, 273
Promedas, 97–100

RAAM, *see* recursive auto-associative memory
random indexing, 300–301
random labeling, 301, 304
rate probabilistic functions, 231
Real World Computing program, 3
real-world intelligence, 2
recombination, 125–130, 133, 134, 143, 170, 184
recursive auto-associative memory, 254, 269
reduced representation (*see also* holographic reduced representation), 254
regret, 227
relative entropy loss, 213
replicator equation, 176
  discrete diversified, 177–182
response, 126, 135, 140, 143, 144, 149, 161, 162
response variate, 40
Robbins's proportions, *see* linkage equilibrium

robustness, 257, 272, 274, 277
Royal Road, *see* functions
rule-following, 257, 258, 271
running intersection property, 158
RWC partnership, 3

S-Fixed-Share-Update, 222
S-Loss-Update, 214
Saw, *see* functions
scaling, 254, 274–276, 283–289
schema, 125, 130–132, 147
SDL, 212
SDM, *see* sparse distributed memory
SDS, *see* standard deviation schedule
selected-coordinate design SDM, 274, 283, 290–293
selection, 128, 133, 164, 167, 176, 184
  Boltzmann, 154, 155, 159, 162, 169, 170, 178
  proportionate, 126, 129, 130, 132, 134, 139, 143, 145, 149, 161, 167, 180
  proportionate,, 125
  tournament, 144, 146
  truncation, 140, 144–146, 150, 161, 167, 180
semantic knowledge, *294*, 305–308
semantic similarity, 298
semantic vector, 294, 299–302, 306, 307
semantically similar, 302, 303, 307
SGA, *see* simple genetic algorithm
Sherrington-Kirkpatrick model, 84
sigmoid belief network, 74, 89, 91
signal-to-noise ratio, 275, 284, 286, 293
simple genetic algorithm, 125, 130, 147, 159, 170, 184
socially embedded learning, 27
sparchunk code, 272–279
sparse coding, 254, 272–279, 285, 301
sparse distributed memory, 273–276, 279, 283–286, 289–293
spatter code, 254, 255, 269, 272, 273, 276
specialist decision list, 212
specialist model, 211
square loss, 213
standard deviation schedule, 161, 165, 168, 170
state variate, 40
static parametric mapping, 49
stimulus variate, 40
stochastic approximation, 230
SWML, 214
synonym test, 302, 308

TAP approximation, 75
tempo tracking, 100, 101
tensor-product binding, 254
the apple tasting model, 228
thinning, 273, 276, 277
  example of, 278
Turing Machine, 252
  as a model of the brain, 252

UMDA, *see* univariate marginal distribution algorithm
univariate marginal distribution algorithm, 134, 139, 141, 143, 145, 147, 150, 151, 168, 174, 175, 178, 183, 184
universal emulator, 252
Universal Turing Machine, 252

variance, 126, 140, 141, 143, 144, 149, 152, 161, 163
  additive genetic, 135, 143, 152
variational approximation, 74, 77, 79

WISARD, 293
Wittgenstein's theory of meaning, 294, 298
Wright's equation, 134, 136–139, 166, 176–181, 183–185