Generativity and Systematicity in Neural Network Combinatorial Learning

Olivier Brousse
Department of Computer Science &
Institute of Cognitive Science
University of Colorado
Boulder, CO 80309–0430
olivier@cs.colorado.edu

University of Colorado at Boulder

Technical Report CU-CS-676-93

October 1993

Generativity and Systematicity in Neural Network Combinatorial Learning

Abstract

This thesis addresses a set of problems faced by connectionist learning that have originated from the observation that connectionist cognitive models lack two fundamental properties of the mind: Generativity, stemming from the boundless cognitive competence one can exhibit, and systematicity, due to the existence of symmetries within them. Such properties have seldom been seen in neural networks models, which have typically suffered from problems of inadequate generalization, as examplified both by small number of generalizations relative to training set sizes and heavy interference between newly learned items and previously learned information.

Symbolic theories, arguing that mental representations have syntactic and semantic structure built from structured combinations of symbolic constituents, can in principle account for these properties (both arise from the sensitivity of structured semantic content with a generative and systematic syntax). This thesis studies the question of whether connectionism, arguing that symbolic theories can only provide approximative cognitive descriptions which can only be made precise at a sub-symbolic level, can also account for these properties. Taking a cue from the domains in which human learning most dramatically displays generativity and systematicity, the answer is hypothesized to be positive for domains with combinatorial structure.

A study of such domains is performed, and a measure of combinatorial complexity in terms of information/entropy is used. Experiments are then designed to confirm the hypothesis. It is found that a basic connectionist model trained on a very small percentage of a simple combinatorial domain of recognizing letter sequences can correctly generalize to large numbers of novel sequences. These numbers are found to grow exponentially when the combinatorial complexity of the domain grows. The same behavior is even more dramatically obtained with virtual generalizations: new items which, although not correctly generalized, can be learned in a few presentations while leaving performance on the previously learned items intact.

Experiments are repeated with fully-distributed representations, and results imply that performance is not degraded. When weight elimination is added, perfect systematicity is obtained. A formal analysis is then attempted in a simpler case. The more general case is treated with contribution analysis.

Acknowledgments

I wish to express my gratitude to Mike Mozer, as well as Clayton Lewis, Michael Main, and Kelvin Wagner, for insightful conversations about this research. Thanks also to the members of the Boulder Connectionist Research Group, and the Computer Science Systems Group at the University of Colorado at Boulder, for the first rate research and computing environment.

This work owes its existence to an original idea from Paul Smolensky. In addition, his contributions permeate the concepts formulated here. While I carry full responsibility for the views expressed, the critical role played by Paul is fully and gratefully acknowledged.

This technical report is a formatted version of my thesis. This work has been partially supported by NSF grant IRI-8609599.

Contents

List of Figures

List of Tables

Chapter 1

Nature of the problem

1.1 Overview of this chapter

The problem addressed by this thesis is presented in this introductory chapter. After a brief review of artificial neural networks and connectionism – the attempt to model cognition, or aspects thereof, with artificial neural networks, it is noted that while the first artificial neural networks appeared more that fifty years ago, their pervasive intrusion into the field of cognitive science is much more recent. It is only within the last half decade that they have been recognized within the cognitive science and artificial intelligence communities as a serious framework for cognitive modeling as well as an alternative to the traditional approach presupposing a symbolic ontology. In the context of this widespread recognition connectionism is evaluated against the traditional artificial intelligence approach. It is realized that while it has seemed to be an adequate tool for modeling certain low-level cognitive tasks, it has been less successful in modeling some fundamental properties of cognition displayed in highly conceptual cognitive domains such as language. The question of whether there is no obvious and fundamental limitation of the neural computation paradigm which would prevent it, in theory, from modeling such properties is therefore asked. Connectionism, clearly, needs to address this serious issue if it is to be considered as a viable framework for cognitive modeling. The question is placed in context by presenting the problem from an artificial intelligence perspective, as it is noted that these properties have been more easily addressed by the traditional symbolic artificial intelligence school.

A number of such basic properties displayed by the human mind are examined, namely generativity of thought, systematicity and compositionality of mental representations, and systematicity of inferential processes, properties that have been described in (Fodor and Pylyshyn, 1988). The question of how the connectionist framework fares with respect to their modeling is addressed, leading to the conclusion that experiments which could convince observers believing that some of these properties cannot in principle be neurally modeled are needed. The connectionist theory of concepts, due to Cussins (1990) is then examined. This general theory makes the point that connectionism is not, in principle, restricted in such a way that these properties could not be modeled.

The last section of the chapter, finally, presents a refutation of the critique of connectionism found in (Fodor and Pylyshyn, 1988).

1.2 Goal and overview of this thesis

Chapter 1 comes to the conclusion that connectionist systems would fail as proper cognitive models if they could not *in principle* allow:

(P1) Compositional representations

(P2) Systematic representations

(P3) Structure-sensitive processing

(P4) Generative processing

Examining current connectionist research, we show that distributed representations are a case of compositional representations (**P1**), that tensor product and RAAM distributed representations are a case of systematic representations (**P2**), and compositional representations (**P1**) a fortiori.

The goal of this thesis is to then show experimentally that structure-sensitive processing (**P3**) and generative processing (**P4**) can exist in connectionist networks, if it involves the connectionist processing of representations which are instances of compositional representations (**P1**) and systematic representations (**P2**). We will, in addition, experimentally show that with compositional (**P1**) and systematic representations (**P2**), structure-sensitive processing (**P3**) and generative processing (**P4**) can be *learned to be displayed* by connectionist learning procedures. The success of such experiments will in no way give a solution to the general problem of connectionist modeling of the properties listed above. It will however show that connectionism is not in principle incompatible with these properties.

This thesis is divided as follows.

In chapter 2 we present our experimental approach. We chose the most simple combinatorial domain– whose elements present, by definition, compositionality and systematicity, as our experimental domain, as we believe that such domains are the ones were generativity and systematic processing will emerge. We define the connectionist auto-associative apparatus and representational scheme used, and present the cognitive processing tasks modeled (**Inference of regularity due to combinatorial laws in a simple domain consisting of a Cartesian product of sets**). The notion of generalization, central to connectionist learning, is studied, and linked to the problem of interference in learning between newly learned items and previously learned information. An extension of the notion of generalization called virtual generalization, which can be used to measure competence, is introduced. We also review research literature pertaining to the chosen tasks.

In chapter 3, the notion of combinatorial domain is studied. An attempt is made at a characterization of the notion, and a measure of combinatorial complexity, central to the study of generativity, is investigated. An attempt at a mathematical characterization of the kinds of connectionist representations well suited to encode members of combinatorial domains is also made.

Chapter 4 reports on a number of experiments performed with semi-distributed representations. Massive true and virtual generalization, and their explosive growth as the complexity of the domain are increased, is observed. The complexity of the domain, the architecture of the networks used, the learning rule, the sizes of the training sets, and the training criteria are varied.

In Chapter 5, we discuss the relation and differences between semi-distributed representations and fully-distributed representations. The case is made that the use of semi-distributed representations is not crucial to the obtention of the results reported in the previous chapter, and that fully-distributed representations can, in theory, lead to the same results. We also present further experiments implying experimentally that our results are also valid when fully-distributed representations are used.

Chapter 6 repeats a number of experiments reported in chapter 4, with a modified learning rule implementing the technique of weight elimination. Perfect systematicity and generativity are obtained in this case.

In chapter 7, an attempt at a formal mathematical analysis of the weights developed by some of the networks reported on earlier, in a simpler case, is made. Statistical analysis, including cluster, principal component, and contribution analysis are then performed in the general case.

Chapter 8 concludes this thesis.

1.3 Connectionist models

1.3.1 Introduction

The idea that computation can be performed by assemblies of neuron-like elements is hardly new and can be traced back to at least the middle of the 20th century. While the pioneering neural network research of McCullogh and Pitts (1943) was concerned with computational and representational issues, Hebb (1949) introduced learning principles, and Rosenblatt (1958) introduced a network, the Perceptron, that was capable of learning. As later analytical work by Minsky and Papert (1969) showed severe theoretical limitations of the Perceptron, and as the rise of digital computers boosted research in symbolic artificial intelligence, neural network research stayed on a low plateau until the 1980's, when Ackley et al. (1985) and Hinton & Sejnowski (1986) introduced a learning algorithm for Boltzmann machines, Rumelhart et al. (1986a) experimented on a learning algorithm overcoming the limitations of the Perceptron learning procedure, and Rumelhart and McClelland's group of cognitive scientists at the University of California at San Diego (Rumelhart *et al.*, 1986b) (McClelland *et al.*, 1986) demonstrated the power of the connectionist approach for modeling diverse aspects of cognition. This rebirth of connectionism also coincided with the realization that symbolic artificial intelligence as a cognitive modeling tool seemed to be inadequate in many domains, especially those where properties such as robustness to noise, self-organization, graceful degradation, and massive parallelism –all natural properties of neural networks–, were needed.

Since then a tremendous renewal of interest in the use of connectionist networks has occurred, for modeling cognitive processing tasks as diverse as speech, pronunciation, vision, language processing and motor control. ((McClelland and Elman, 1986) (Sejnowski and Rosenberg, 1987), (Zemel *et al.*, 1990), (Elman, 1990), (Jordan and Jacob, 1990), for instance)

1.3.2 A brief description

Connectionist (or neural) networks are artificial networks whose architecture and governing principles are inspired by the biological networks of neurons found in the brain. They usually consist of a large number of very simple processors, each characterized by a real-valued activity, interconnected by lines capable of transmitting activities to a degree depending on their strength or weight. The pattern of connectivity among processors of a connectionist network define its architecture, varying from one model to another. In some models, all units are inter-connected to every other unit of the network. In others, connections only exist between units of different layers, or pools of units. Each processor of a connectionist network is typically able to compute a simple function (activation function) of the weights of the lines and values of the activities of the processors it is connected to.

Input to the system will be patterns of activity distributed over those units of the network chosen to represent input patterns (the input layer) and output from the system will be the activities of those units that have been chosen to represent output, conceptually grouped together as output layer. Processing in connectionist networks consists in applying input patterns to the input layer, letting activity flow, via the weighted lines, from the input processors to all connected processors in the network (including processors that are not used for input nor output–these are the hidden units). Output patterns are then obtained at the output layer. Processing can thus be seen as the evolution, through time, via the weighted lines, of a pattern of activity over the set of processors.

The values of weights are typically learned with a learning algorithm that is applied before testing is performed. In supervised learning, both desired input and output patterns are applied to the input and output layers, respectively, and the network weights are iteratively modified according to the learning rule used to yield, at processing time, the correct association. In unsupervised learning, only input patterns are presented, and connections are modified to produce at the output layer some interesting function of the input pattern. A number of different learning rules have been used to modify the weights of a connectionist network to ensure proper behavior at processing time. These learning rules usually implement a variant of the principle according to which the strength of a connection is increased or decreased in the direction of correct processing.

1.4 A viable framework for cognitive modeling?

1.4.1 Introduction

While most connectionist research has been concerned with finding and implementing new architectures, processing algorithms, and learning rules which would allow for an adequate modeling of specialized cognitive tasks, recent years have also witnessed principled *analyses* of connectionism as a paradigm for modeling cognition (Hofstadter, 1985) (Derthick, 1986) (Dreyfus and Dreyfus, 1988) (Fodor and Pylyshyn, 1988) (Smolensky, 1988). These analyses have had several merits.

First, they have reflected on the most general and theoretical question of the possible adequacy of connectionism for modeling cognition. They have addressed, from a connectionist point of view, some basic properties of mind which need to be accounted for if connectionism is to be viewed as a viable paradigm for modeling cognition, and tackled the fundamental problem of whether there are theoretical, in principle, restrictions which would prevent it from allowing for an account of such properties. These reflections are indeed very valuable: While solving punctual problems advances one's knowledge of the field and is often the only possible activity allowing concrete results in a task so vastly complex, identification of larger problems and reflection about the very adequacy of the framework for solving them is a much needed endeavor if the paradigm is to have any future, especially when it has come of age.

Second, they have situated neural network modeling within the general field of cognitive modeling. Indeed, connectionism, as a cognitive modeling tool, did not enter a field in its infancy. The questions of what makes up a mind, of what are its general governing principles and properties which need to be taken into account if one is to model intelligent behavior, are questions which have been studied for several centuries by philosophers of mind, and at least for several decades by artificial intelligence researchers and cognitive psychologists alike. It is therefore important that connectionism, as a computational framework targeted at providing the building blocks with which to model cognition, situate itself within the general field of cognitive modeling. Any reflection on the possible adequacy of connectionism to model cognition, any enquiry about the possibility that connectionism might be the right modeling paradigm, needs to benefit from an evaluation of the paradigm against the existing standards. It is the subsequent uncovering of incompatibilities and/or limitations that will allow us to define the necessary starting base of problems upon which a remedying research plan can be developed.

It is problems emerging from such analyses, as well as related independent observations of our own, which prompted the line of research presented in this thesis. In the next sections we present a historical and philosophical perspective on both symbolic and connectionist artificial intelligence. This perspective will provide the needed background and context in which these analyses and the problems they have articulated have evolved. We will then more specifically define the problems addressed by this thesis.

1.4.2 Symbolic artificial intelligence

Before the widespread appearance of neural networks for cognitive modeling, the world of artificial intelligence was dominated by researchers belonging to what we might dub the "symbolic" school. One of its most predominant figures, Allen Newell, formalized in his influential paper, "Physical Symbol Systems" (1980), the principles that have driven symbolic artificial intelligence research in the last three decades.

In that paper, Newell articulates a hypothesis, according to which:

> The necessary and sufficient condition for a physical system to exhibit general intelligent action is that it be a physical symbol system. Necessary means than any physical system that exhibits general intelligence will be an instance of a physical symbol system. Sufficient means that any physical symbol system can be organized further to exhibit general intelligent action.

Defining a physical symbol system to be the same as a Universal Turing machine, Newell thus hypothesizes that intelligence can be replicated to any degree of complexity in a sufficiently complex symbol system. That school of thought has been shared by a number of prominent cognitive scientists, among which stand Fodor and Pylyshyn (1975) (1988), who have argued, as we will see later, for a mind governed by a "language of thought", where elemental entities are symbols. Intelligence would thus originate from manipulation of symbols according to the "grammar" of the language of thought.

The past four decades has witnessed tremendous research activity aimed at developing computer programs modeling intelligent behavior, assuming the physical symbol systems hypothesis. The serial and discrete traditional Von Neumann computers, no doubt, were and are well suited to model and implement these "symbolic" views of the mind. But the passing decades produced a growing awareness that the symbolic AI models are limited as far as replicating intelligence was concerned. Certainly, very sophisticated schemes have been developed and used to code and process knowledge in these programs: semantic networks, for instance, where semantic dependencies of knowledge symbols are links in a network representing a memory. Schemas, where memory is assumed to be organized in terms of many small packets of related knowledge. Or frames, where units of structured knowledge are hierarchically linked and contain procedural knowledge as well. But these programs have been characterized as inflexible and brittle (they break when asked to work beyond the borders of the domain that was formally described at implementation time), unrealistically slow (their internal organization sometimes imply lengthy search at processing time) and unrealistically hard to maintain (limited learning capabilities prevents them to adapt smoothly and naturally to new knowledge or conditions).

As these limitations were becoming more obvious, a different alternative view of the mind started to be proposed, along with a different approach –the connectionist approach– to modeling intelligence. Below we outline one such view, presented in (Smolensky, 1988).

1.4.3 The sub-symbolic approach

The sub-symbolic approach to cognitive modeling starts by acknowledging the existence of two kinds of knowledge. A line is drawn between one kind called cultural knowledge, which appears to be governed by conscious rule interpretation, and another which encompasses individual knowledge, skill and intuition. It is hypothesized that the virtual machine running the program responsible for behavior that is not conscious rule interpretation, called the intuitive processor, has a connectionist architecture: It is a dynamical system whose entities are complex patterns of activities, encoding sub-symbolic knowledge, that does not admit a complete, formal and precise conceptual (symbolic) description. A precise description of the intuitive processor can only be made at the sub-conceptual level. That level is different from the brain level, where the intuitive processor is actually implemented. Intuitive knowledge which appears to be rule following only reflects the large scale structure of micro-computations, and any conceptual description of that knowledge will only be, therefore, an approximation. Identifying the human cognitive machine as a large dynamical system that can only be precisely described at the sub-symbolic level has a number of important consequences. In the subsymbolic paradigm, symbols are not atomic entities but collections of sub-symbols. Their context reside in the symbol itself, in the subconceptual entities the symbol is made of. Knowledge resides in the connections, and inference is a massively parallel process.

The connectionist paradigm, then, states, in opposition to Newell, Fodor and Pylyshyn, that intelligence should be described and analyzed at the sub-symbolic level, a level lower than the symbolic one described by Newell, yet higher than the implementation level where neurophysiological transformations occur. The symbolic behavior observed by the researchers mentioned above is only an approximate macroscopic description of complex interacting microscopic behaviors. Intelligence does not occur with symbol manipulation, but with activation flows in large numbers of connectionist units.

1.4.4 Compatibility of the two approaches

It is important to notice that the sub-symbolic approach refutes the physical symbol system and the language of thought hypotheses, by replacing symbols by connectionist units as the atomic entities needed to describe the mind, but pushes for the view that symbolic and connectionist accounts are compatible once it is understood that one is an approximation of the other. A natural question that one might ask, then, if one views the connectionist approach as a serious alternative to the symbolic approach, is the following:

(1) Can connectionist models account for properties most naturally obtained by symbolic models?

In other words, can the former do as well where the latter is best, that is, in domains that can best be described by symbolic manipulation, such as language processing? Surely, if symbolic descriptions are correct approximations of cognitive behavior which can only be precisely accounted for at a connectionist, sub-symbolic level, then the natural properties found in symbolic accounts should emerge from connectionist accounts. Just as, for instance in the field of thermodynamics, the macroscopic and approximate law that relates the pressure, the volume and the temperature of a perfect gas, $PV = nRT$, can be derived from the more precise microscopic laws governing the momentum and kinetic energy of individual molecules of the gas, if one makes the necessary approximations, symbolic properties of physical systems or a language of thought should be derived and emerge from connectionist accounts.

A brief look at the connectionist literature would not, we believe, convince anyone of either a positive or negative answer to question **(1)**, [1] although it would certainly induce a definite bias in favor of a negative answer. Indeed, it is interesting to note that whereas connectionism often fares relatively well where the symbolic approach has consistently had severe problems, in areas such as ill-defined problem solving, massive constraint satisfaction or noise-filled environment processing, it has not been able to come even close to replicating the modeling power of symbolic systems when it comes to addressing certain basic properties of cognitive behavior. A brief analysis of the symbolic and connectionist approaches to language acquisition, in

[1] Although Fodor and Pylyshyn have certainly been convinced of a negative answer, wrongly as we will later find out.

the remainder of this section, will convince us of this fact.

A case in point: Language acquisition.

Human language acquisition is certainly a definite example of high competence acquisition from a small set of examples. Although how much competence one believes is acquired depends both on how powerful one deems the inductive apparatus is and on how much information is conjectured to be available during learning, it would be hard to find anyone who would feel remotely comfortable with the claim that the set of examples children are trained on is of the same order of magnitude as the set of examples they ultimately generalize to.

Symbolic accounts offer a simple and most natural explanation of how human language acquisition allows for the possibly unbounded generation and understanding of language constructs from a relatively small and finite set of examples:[2] they point out that a system composed of atomic elements (symbols) and combinatorial laws governing their composition (grammar rules) will display a property of generativity, since the number of valid constructs grows explosively with the number of atoms and combinatorial laws.

Sub-symbolic accounts, on the other hand, have not so far seemed to be able to offer a similar explanation. Although learning from examples is a well-known characteristic of connectionist networks, they seem to be at odds with the concept of generativity. A passage in one of the earliest works in neural networks ((Rosenblatt, 1962), page 73, emphasis is ours), is especially revealing:

> In a simple perceptron, patterns are recognized before "relations"; indeed, abstract relations, such as "A above B" or the triangle is inside the circle" are never abstracted as such, **but can only be acquired by means of a sort of exhaustive rote-learning procedure**, in which every case in which the relation holds is taught to the perceptron individually.

And indeed, even current models using more powerful learning techniques than the perceptron suggest that in order to obtain correct performance on a target set of inputs, a network needs to be trained on a sizable fraction (between 25% and 75%) of the learning set. (e.g. (Hinton, 1987), (Saito and Nakano, 1988), (Le Cun *et al.*, 1990)).

In light of this example, then, and abstracting over many other tasks tackled both by the symbolic and the sub-symbolic approaches, we believe that question **(1)** has not been directly and experimentally addressed.

Our approach to addressing question **(1)** can now be described. We will consider several properties described by an approximate symbolic theory of mind, and attempt to show that connectionist models can also provide an account. Below we review, following (Fodor and Pylyshyn, 1988), four general and fundamental properties of symbolic processing found in cognition. It is our opinion that these properties in no way characterizes cognition exhaustively. But as was mentioned before, we also believe that connectionism would fail as a plausible framework for modeling cognition if it exhibited inherent restrictions which would prevent it from accounting for them.

It is the central concern of this thesis to address the theoretical question of whether these properties can *in principle* be modeled by connectionist networks. After reviewing them, we will see how connectionist research has addressed and/or modeled them so far. We will point out deficiencies, then investigate what can be done to show that there is no in principle fundamental restriction which would prevent us from explaining and modeling them.

1.5 Connectionism and some fundamental properties of cognition

1.6 Descriptions

The following four properties of the mind were put forward by Fodor and Pylyshyn (1988) and are here described.

Generativity of thought

The classic argument for the generativity of thought revolves around the fact that we can produce a potentially

[2] We will present this explanation in greater detail in the next section.

unbounded number of mental representations under finite means. This in turn pushes for the assumption that mental representations are expressions combined from atomic elements according to combinatorial rules, thus allowing for the view that the set of possible representations is a *generated* set. The laws of combinatorics, inducing an explosive growth in the number of possible combinations as the number of atomic elements increases, allow for the extremely large (potentially unbounded) number of combined representations. This reasoning was originally offered for language, e.g. (Chomsky, 1968), which exhibits an obvious generative capacity: we can generate or understand an extremely large number of sentences, the majority of which have not been uttered or heard before. The view that sentences are generated from the combination of words of the lexicon according to grammar rules explains the unboundedness of grammatically well-formed sentences.

Systematicity of representations

Systematicity seems an inherent property in language production and language understanding when it is noticed that understanding or producing the sentence "John loves Mary" could not possibly go without understanding or producing "Mary loves John". As Fodor and Pylyshyn state, "The ability to produce or understand some sentences is intrinsically connected to the ability to produce/understand others." ((Fodor and Pylyshyn, 1988), page 37). The argument for the systematicity of thought follows from the one for the systematicity of linguistic capabilities by observing that just as understanding "Mary loves John" entails understanding "John loves Mary", the ability to have the *thought* of the first expression (which necessarily precedes its utterance) entails the ability to have the thought of other. The systematicity of thought makes the case for the internal structure of thought.

Compositionality of representations

Closely related to systematicity, the "principle of compositionality" as stated by Fodor and Pylyshyn is based on the fact that, in language, sentences that are systematically related are not arbitrary from a semantic point of view. Syntactical constituents of sentences that are systematically related must be composed in each of the sentences to have approximatively the same semantic contribution. The argument about sentences in language can also be made concerning mental representations in the language of thought. From this it follows that one needs to assume that mental representations have both syntactic and semantic compositional structure.

Systematicity of inference

Systematicity of inference is a property illustrated by the simple following example: If one can make the inference from P&Q&R to P, one must be able to make the inference from P&Q to P. As Fodor and Pylyshyn state, "inferences that are of similar logical type ought, pretty generally, to elicit correspondingly similar cognitive capacities." This is yet another argument for the constituent structure of thought and for the fact that processes operate with respect to that structure. If mental processes did not operate on that structure, it would be possible to make one inference but not another on two expressions which had the same structure.

We now look at the properties just defined and investigate whether they have been displayed within the connectionist framework.

1.6.1 Compositionality and connectionist representations

Can connectionist representations be compositional? That is, can connectionist representations be built up, or composed, from the connectionist representations of constituents? The answer, obviously, depends on the kind of connectionist representation used, which in turn depends on how the problem of connectionist representations –How can concepts (possibly structured) be mapped to activity patterns over a set of connectionist units? –is solved. Below we therefore review connectionist representational schemes, and present an argument, "the coffee story", that answers the question in the affirmative, for the case of distributed representations.

Local representations.

A simple solution to the connectionist representation problem is to simply devote one unit of the network per concept (local representation). Limitations of this scheme are obvious: Since relations among representations of concepts can only be implemented in the connections of the network, it is the *way* activation flows in the network that regulates relationships among concepts. And it will not reflect, in general and except in the most trivial cases, the relationships desired, unless each connection is specifically hand-tuned to induce the desired effects in activation flow.

Likewise, since concepts with constituent structure can only be coded (by definition) by assigning a single unit for each constituent, structural relations among constituents can only reside in the connections and are regulated, again, by the way activity flows. But as seen before, this regulation will not be the desired one in general.

Local representations, therefore, are not suited to display compositionality.

Distributed representations.

Another solution to the connectionist representation problem is to represent concepts over *sets* of units. If each concept is represented over a distinct set of units, the representation is called semi-distributed. If concepts are represented on a single set of units where each unit participates in the representation of many concepts, the representation is called fully distributed.

The coffee story: Distributed representations can have compositionality

In (Smolensky, 1987a), Smolensky makes the case for the compositionality of distributed representations. Let us suppose that we have a representation of the structured concept *cup with coffee* (with constituents *cup* and *coffee*) using micro-features. The concept will thus be represented in a distributed way over a set of units, where each unit corresponds to a microfeature (such as "is an upright container" , "has a burnt odor", "brown liquid contacting porcelain", ...) and is activated when that microfeature is present in the description of the concept.

Is there a compositionality relation between the representations of *cup with coffee* and the individual representations of the constituents *cup* and *coffee*? Yes: if one subtracts the representation of *cup without coffee* from the representation of *cup with coffee*, one obtains a representation of *coffee*, in the context of being in a cup. Compositionality exists, therefore. Note that the fact that the representation obtained from the subtraction above corresponds to coffee *in the context of a cup* shows that the compositionality is approximative: the result is not the representation of the context-independent concept of coffee, yet is close to it. Note also that the compositionality stands at the level of the patterns of activities. Also, in the case of distributed representation, relations among concepts or structural relations among constituents of concepts are coded directly *in the patterns of activities*. This is quite different from local representations where *connections* were the coding agent.

1.6.2 Systematicity of representations

We have seen above that distributed representations can allow for compositionality. In this section we present two representational schemes which allow for systematicity. That is, they allow two or more representations of constituents to have the same structural role with respect to the representation of the structured concept they belong to.

Tensor product representations

The basic idea of the tensor product representational scheme, developed by Smolensky (1987b), (1990), is the following: Any concept with a constituent structure, made of constituents combined together according to the structural role they play in the concept, can be represented by combining the representations of each constituent and its associated structural role.

More specifically, coding of a structured concept can be performed by:

1. Finding a representation for each of the constituents,
2. Finding a representation of the structural role each one has,

3. Performing the tensor product operation[3] of the representations of each constituent with the representation of its structural role,

4. And adding the representations of the pairs (constituents/associated structural role) together.

A tensor product representation of the string AB, for instance, can be formed by:

1. Finding representations for the constituents A and B. Say (1,1,0), (1,0,1), respectively, where (1,1,0) and (1,0,1) are vectors of activities of the three units used to code the constituents.

2. Finding representations for the structural roles of each constituents, here their position in the string. Say r1 = (1,1) for the structural role of A, r2 = (1,-1) for the structural role of B.

3. Performing the tensor product of each pair constituent/structural role representations: A $\otimes$ r1 = (1,1,0,1,1,0) and B $\otimes$ r2 = (1,0,1,-1,0,-1).

4. Adding the pairs together which will lead to the representation A $\otimes$ r1 + B $\otimes$ r2 = (2,1,1,0,1,-1) for the string AB.

Note that the tensor product representational scheme can be used, recursively, to code higher level structured concepts from lower-level ones, by applying the same algorithm described above bottom-up.

"Induced" distributed representations: Pollack's RAAM

The "Recursive Auto-Associative Memory" (RAAM) representational scheme, due to Pollack (1988), can be used to represent symbolic structures, and, like the tensor product scheme, allows for systematicity of representations. Unlike the former, however, it requires, as we will see below, learning. The scheme is here presented in the context of representing structured symbolic trees.

RAAM makes use of the following idea: Starting at the base of the tree to be represented, with a representation of each terminal leaf of the tree (where each leaf is represented by a pattern of activity over n units), a compressor can be used to sequentially compress the representation of each set of leaves belonging to the same sub-tree to a representation over n units. The compressor can be used recursively at each level of the tree, from the base to the top, compressing the intermediate representations of each subtree at that level, until the whole tree is represented over n units. To decompose a tree into its original constituents, a decompressor can be used which will, recursively and from top to bottom, produce the intermediate representations obtained in the compressing phase.

The compressor/decompressor used by Pollack is a connectionist network with three layers associating each pattern to be compressed with itself, and the compressed representations are the patterns of activity obtained on the hidden layer (which has n units) when the patterns to be compressed have been correctly auto-associated through learning. The decompressor is the connectionist network which will produce, from a compressed pattern, the decompressed one: it is thus the two layer network made of the hidden and output layer of the compressor/decompressor network. The compressor, producing a compressed pattern from an input pattern, is the two layer network made of the input and hidden layer of the compressor/decompressor network.

Again, as was the case with tensor product representations, the RAAM representational scheme allows the connectionist representations of constituents to have structural roles with respect to the connectionist representation of the structured concept they belong to.

1.6.3 Systematicity of inference

Systematicity of inference, we believe, has been displayed by connectionist networks, albeit in a very weak form: Correct processing of representations is usually the case when these representations are similar (in terms of Hamming or Euclidean distance) to ones the network has been trained on. This systematicity of inference is weak because it does not involve the processing of structured compositional and systematic representations

[3] The tensor product of a p-dimensional vector x with a q-dimensional vector y, $x \otimes y$, is the outer product of the two vectors, which is the $p \times q$-dimensional matrix (or vector) whose elements $M_{i,j}$ are $x_i \times y_j$

where correct processing would result both from the correct processing of constituents and attention to the structural role these play within the overall structure. That kind of strong systematicity has, until very recently , been rarely displayed. [4]

1.6.4 Generativity

Generativity, where a *large* number of representations are correctly processed, relative to the number of representations that have been trained to be correctly processed, is a property that has not, to our knowledge, been instantiated in the connectionist research. It should be noted that generativity is a property that can only appear if the small number of training instances display systematicity and compositionality. It is **these** properties which will allow, through recombination, for the correct processing of a large number of untrained instances. In typical connectionist modeling, as was mentioned in section 1.4.4, a large proportion of representations of objects in the domain to be processed is used for training.

The main points and conclusions made so far, anticipated at the beginning of this chapter (section 1.2 "Goal and overview of this thesis" are now reiterated. We have identified the hypothesis of the sub-symbolic modeling paradigm, that descriptions of cognitive behavior can only accurately be made by connectionist accounts; that symbolic accounts are only be approximation. We have realized that connectionism has so far seemingly failed to address or replicate fundamental cognitive properties naturally accounted for by the symbolic paradigm. In particular, we have come to the conclusion that connectionist systems would fail as proper cognitive models if they could not *in principle* allow:

(P1) Compositional representations

(P2) Systematic representations

(P3) Structure-sensitive processing

(P4) Generative processing

Examining current connectionist research, we have seen that distributed representations are a case of compositional representations (**P1**), that tensor product and RAAM distributed representations are a case of systematic representations (**P2**), and compositional representations (**P1**) a fortiori.

The goal of this thesis can now be made clear in light of this summary: Show experimentally that structure-sensitive processing (**P3**) and generative processing (**P4**) can exist in connectionist networks, if it involves the connectionist processing of representations which are instances of compositional representations (**P1**) and systematic representations (**P2**). We will, in addition, experimentally show that with compositional (**P1**) and systematic representations (**P2**), structure-sensitive processing (**P3**) and generative processing (**P4**) can be *learned to be displayed* by connectionist learning procedures. The success of such experiments will in no way give a solution to the general problem of connectionist modeling of the properties listed above. It will however show that connectionism is not in principle incompatible with these properties.

1.7 The connectionist construction of concepts

1.7.1 Introduction

Before closing this chapter and presenting the approach that will be used to unearth instances of the properties of generativity and systematicity in a connectionist model, it is important to ask just how such properties could possibly arise. While the point was made earlier that connectionist representations can be compositional and systematic, and thus in theory lead to systematicity and generativity, the problem of how typical connectionist **learning** with such representations could lead to systematic and generative behavior is open: For such behavior presupposes that the network has learned to develop, from specific training instances only, recognition of

[4] Recent independent work by Chalmers (1990a), has addressed (**3**): Using Pollack's RAAM representational scheme, Chalmers has shown that a network processing structured sentences which were represented with a non-concatenative representational scheme (see next section) was sensitive to the internal structure of the sentences.

More recent research in connectionist linguistics has also reported systematicity of processing (Legendre *et al.*, 1990a) (Legendre *et al.*, 1990b) (Legendre *et al.*, 1991) (Elman, 1991).

structural constraints [5] in such a way that any new object of the domain, just because it obeys these structural constraints, is correctly processed. This last point might best be illustrated by a simple example: Systematicity of understanding, as seen in the fact that one always understands the sentence "Mary loves John" if one understands the sentence "John loves Mary", for example, originates from the fact that both sentences impose structural constraints on the objects they are composed of, that both Mary and John are recognized to be of the same category (noun phrase) and can freely be swapped as objects or subjects. Any cognitive system recognizing that any proper nouns of human beings fall in the same category than the category as "John" and "Mary" will then have no problem explaining that example of systematicity. But how can a connectionist network, using structured connectionist representations like RAAM or tensor product representations, which would only see **specific instances** of "name1 Loves name2", develop competence in such a way that **any** pattern of the form "name1 Loves name2" would be correctly processed?

A recent enlightning paper by Cussins, exceptional and ground breaking in my opinion, revisits the symbolic and connectionist paradigm from a philosophy of mind perspective and throws some light on an answer to the above question. Its most pertinent points are first reviewed below [6].

1.7.2 Cognitive explanation and the problem of embodied cognition

In his paper, Cussins defines what he calls the problem of embodied cognition, which is the central problem that any cognitive science theory, attempting to explain how physical systems can think, needs to solve: How can neurophysiological explanations at the brain level and cognitive explanations at the cognitive level, each autonomous, relate and go hand in hand?

Such a problem, the case is made, needs to be solved without reducing cognitive properties to non-cognitive properties, nor eliminating cognitive properties, nor rejecting the scientific indispensability of cognitive properties. The problem of embodied cognition is then given a necessary and sufficient condition for its solution.

This problem can be solved if and only if, Cussins shows, the cognitive theory explains how cognitive properties at a high level can be constructed out of non-cognitive properties at a low level. Such an explanation is possible if in between the highest and lowest level, there exists a number of intermediate levels such that, although explanation and observation of a cognitive phenonema seem autonomous and unintelligibly related in between the two extreme levels, it is intelligibly related between any pair of adjacent intermediate levels.

1.7.3 Conceptual content and the language of thought

Cussins makes the point that the language of thought cognitive theory, a theory of cognition at the level of psychological explanation in terms of conceptual properties due to Fodor (1975), solves the problem of embodied cognition: the required intelligible link between the computational component [7] and the psychological component of modeling in cognitive science is made by the development of a syntactic theory and semantic representational system, for which the syntax is implemented computationally and the semantics is suitable for psychological explanation. The syntactic theory of the representational system, allowing for a definition specification of all the legal combinations of the atomic representations of the system, thus marches in step with the semantic theory, which provides specifications of the interpretation of all the representations.

1.7.4 Non-conceptual content and the connectionist construction of concepts

The language of thought theory is a theory involving a semantic theory of **conceptual** content: content which presents the world to a subject as divided up into objects, properties and situations that have truth condition properties. Cussins makes the point, however, that non-conceptual content, where the world is perceived without being decomposed into objects, properties, or situations, also exists within the cognitive

[5] Both properties presuppose the existence of structural constraints on the constituents of the objects of the domain at hand, governed by combinatorial laws.

[6] The following presentation is intentionally simplified, as a more rigorous and precise description of the connectionist construction of concepts theory of Cussins, laid out by its author in highly technical and philosophical terms, would be too lengthy and fall outside the scope of this thesis.

[7] How the link between the physical substrate, the brain that is, on which the computational component is implemented is carried out seems to remain a mystery, at least for me.

realm. He gives several examples, including some within the linguistic paradigms, of cognitive experiences involving non-conceptual content [8].

From there, it is observed that certain kinds of non-conceptual content entail a lack of objectivity in the subject experiencing it. A link is made between lack of objectivity and **perspective dependence**, and it is shown that a theory of non-conceptual content can still explain conceptual experiences if it provides a mean to reduce the perspective dependence of non-conceptual content, via a computational theory of transformation of non-conceptual content. Such a reduction implements a transition from a non-conceptual content (perspective dependent) to a conceptual content (perspective independent), as perspective independence evolves from perspective dependence. How can such a reduction me made? Cussins argues that it involves the formation, or construction, of a cognitive map, which will allow a non-conceptual representation to be treated independently of its associated perspective, and thus just as a perspective independent conceptual representation.

Having made the case that a cognitive theory based on non-conceptual content can solve the problem of embodied cognition and provide a basis for explaining both non-conceptual and conceptual cognitive experiences, the question of whether connectionism can be be the right computational and architectural paradigm for such a theory is then asked. The answer is positive: Connectionism is a non-conceptualist cognitive modeling theory since the external representations that it uses are naturally perspective dependent, and thus non-conceptual. It can explain conceptual content by reducing the perspective-dependence of the content of the representational states of the system through learning. Such a reduction, as was stated earlier, can be performed via the formation of a cognitive map, which in connectionist terms translates in the development via learning of perspective independent internal representations.

1.7.5 Relation to this thesis

The connectionist construction of concepts theory, then, implies that external, non-conceptual, perspective dependent connectionist representations can, through training, contribute to the formation of an internal cognitive map (represented in the weights of the network) out of which perspective independent processing of such representations will emerge. If we are successful in designing connectionist experiments exhibiting generativity and systematicity, such experiments will necessarily involve the formation of a cognitive map. If our models are simple enough, analysis of such cognitive map formation will be possible. In this case, our experiments will have the advantage of presenting a simple yet concrete and analyzable instance of how the connectionist construction of concepts might be realized.

1.8 A reply to a criticism of connectionism

If successful, our approach will also have the advantage of countering one of the most severe criticism made against connectionism. We briefly introduced Fodor & Pylyshyn earlier in this chapter when we introduced and reviewed the four properties of mind they have outlined in their paper: **Connectionism and cognitive architecture: a critical analysis** (1988). In that paper, as the title implies, Fodor & Pylyshyn go far beyond presenting these properties. They actually use them as a starting point for the argument that connectionism cannot be a viable framework for cognitive modeling.

The underlying line of argumentation driving Fodor and Pylyshyn's claim for the impossibility of a connectionist architecture of the mind can be summarized succinctly:

If atomic constituents of mental representations are symbols, and mental processes operate on these symbols, then we have a symbolic language of thought that can allow for the central properties of cognition such as systematicity, generativity and compositionality. They thus posit the existence of a symbolic language of thought where mental representations have constituent structure on which mental processes operate, and claim that:

Standard connectionist representations cannot have a constituent structure, and connectionist processing cannot be structure sensitive **(i)**

Standard connectionism, therefore, cannot be of any interest for cognitive modeling; the only interesting

[8] Among others: the perception of color. Although there exists color concepts, the point is made that color-experience cannot be explained in a conceptualist way.

connectionist models would be non-standard connectionist models that implement a language of thought **(ii)**.[9]

Now what is the rationale for believing **(i)**? Fodor and Pylyshyn's argument is based on the analysis of networks using *local* representations (where each concept is represented by a single node in the network). They show, convincingly in our view, (we drew the same conclusions in section 1.6.1) that such networks cannot exhibit properties of systematicity and compositionality of representations, let alone generativity of representations and systematicity of inference. They then erroneously conclude that *no* connectionist networks can exhibit these properties.

1.8.1 Refutation

There are at least two ways to logically show that Fodor and Pylyshyn are wrong on **(i)** One is to show that there is nothing to support **(i)** in principle, in exhibiting or designing representations that are compositional and systematic, and show in theory that there is no restriction which would prevent connectionist systems from exhibiting generativity and systematicity in inference. This refutation would make the case that **(i)** is pure conjecture, but would not, however, prove that it is necessary wrong.

Another more powerful way is to show *by example* that **(i)** is wrong by designing and implementing networks whose behavior, relying on compositional and systematic representations, displays generativity and systematic inference. This thesis can be seen as an example of this second approach.

Addressing (i) in principle

Here we proceed to refute **(i)** in theory, by simply showing that its premises, the following assertions:

(1) Connectionist representations cannot have compositionality

(2) Connectionist representations cannot be systematic

(3) Connectionist processing cannot be structure-sensitive

(4) Connectionist processing cannot be generative

are either false or so far unprovable. More precisely, we show that assertions **(1)** and **(2)** are false, while there is no reason to believe that assertions **(3)** and **(4)** are true. Since showing that there is no solution so far to a problem does not mean that the problem is not solvable (to show this one would need to show that *any* approach would fail, something Fodor and Pylyshyn have not done since their assertion about the equivalence of local and distributed representations is unsound), this will show that Fodor and Pylyshyn's argument is logically false.

(1) is wrong in theory because of the very existence of distributed representations, (where concepts are represented over a set of units and each unit participates in the representation of many concepts). As we have seen earlier in our discussion of connectionist representations, distributed representations are fundamentally different from local representations, and do allow for compositionality, as the "coffee story" showed. While Fodor and Pylyshyn were right in their conclusion that local representations could not allow for compositionality, they were wrong in assuming that using distributed representations could not change the nature of their conclusion.

(2) is wrong because of the very existence of the tensor product and RAAM representational schemes. These schemes, as we have seen, allow for systematicity of representations. Fodor and Pylyshyn's erroneous conclusion, again, originated from their lack of concern with more sophisticated representational schemes.

[9] It seems that Fodor and Pylyshyn, reluctantly realizing the success of connectionism, had to fit it *at least somewhere* and could not go all the way and call for an abandonment of all related research. They went on to formulate the assertion that connectionism can still be used as an implementation of a language of thought.

Chalmers (1990b) has analyzed this assertion, blending the use of the terms "standard" and "non-standard", which Fodor and Pylyshyn sometimes seem to do. In this case, their assertion is logically unsound, as it is a well known fact that connectionist networks can implement Turing machines, and thus a language of thought. But then, if it is so, they can have structured representations and structure sensitive processing. This violates the truth of their earlier claims.

(**3**), so far, is a conjecture. As we have seen earlier, no connectionist experiments have been made to specifically either prove or disprove the fact that connectionist representations cannot display generativity. As we have mentioned, it is the central concern of this thesis to actually prove, experimentally, that (**3**) is not true.

(**4**), like (**3**) and for the same reasons, is also a conjecture. Its experimental refutation is also the object of this thesis.

1.8.2 Implementation issues: Concatenative and functional compositionality

It is important, at this point, to reflect on the relation that a connectionist system successful at being systematic and generative (through perspective dependence reduction of its input/output behavior, via a cognitive map formation) would have with a symbolic system. Let us suppose, for instance, that a connectionist system learns, given a training subset of a domain, to behave like a given symbolic system implemented with symbolic rules. That is, for a given symbolic processing task, we use connectionist representations to code symbol structures of the symbolic system, and the input/output behavior of the connectionist system can be described in symbolic, or conceptual terms.

If the connectionist system uses local representations, then each unit of the network will correspond to a symbol in the symbolic system and we will have simply implemented the symbolic system with connectionist means. The connectionist system will simply replicate, then, the behavior of the symbolic system and the two will be isomorphic on that subset of the domain the networks was trained on. The connectionist network will, like the symbolic system, exhibit ungraceful degradation, or brittleness, if any of the (input/output) external units coding a symbol do not function properly. One difference, however, will be that the network might not exhibit such behavior if internal units are damaged.

If the system uses semi-local representations, the story is different. Although there will be a one-to-one correspondence between each pool of units used for the representation of a symbol and that symbol in the symbolic system, the system will not be "aware" of this correspondence,[10] but will have **learned** it. In that case, then, the connectionist system will have learned the spatial correspondence between symbols and the pools of units used for their representation, and will learn to behave like the symbolic system. A description in symbolic terms will be possible, but a precise description of just how the processing mechanism came to be implemented through learning, by the very fact that learning occurred at the level of individual units, will only be possible at the level of individual units and activities of the connectionist system, and not at the symbolic level. Because of the spatial one-to-one correspondence between a pool of units devoted to the representation of a symbolic structure and that symbolic structure, however, it would still be possible to claim that the connectionist system has simply implemented the symbolic system, within an isomorphism. But noise resistance and graceful degradation, on any unit now, will distinguish the network from its equivalent brittle symbolic system. Note also that learning might never allow the network to "perfectly" replicate the behavior of the symbolic system, although it might allow it come close. In this case, the two systems will not be isomorphic.

If fully distributed representations are used, the story might be different again. Van Gelder (Van Gelder, 1990) has analyzed fully connectionist distributed representations and has made the important distinction between concatenative compositionality, which is displayed by structured symbolic expressions, and functional compositionality, displayed by fully distributed connectionist representations. Indeed, Van Gelder notices that any symbolic representation of an expression literally contains the representations of the symbolic constituents it is made of. The structural relation of a constituent and the expression it belongs to is only reflected through its position in the representation of the expression. The function mapping the representation of a constituent to the representation of the expression it belongs to is thus a simple concatenation. In the case of connectionist fully distributed representations, however, the relationships between the constituents' representations and the representation of the structured concept they belong to are not spatial, but functional: A complex function, not merely a concatenation, (composed of several tensor product operations and an addition in the case of tensor product representations) maps the representation of a constituent to the representation of the structured concept it belongs to.

[10] Unless the architecture of the connectionist system is designed to specifically take into account that one-to-one correspondence.

This distinction has important consequences: a fully-distributed connectionist representational scheme using non-concatenative compositionality to code structured representations might not be isomorphic to any representational scheme (like one used in a symbolic system) using concatenative compositionality. Just as in the case of semi distributed representations, then, a connectionist system learning to behave like a given symbolic system will do so by implementing a correspondence, possibly approximate, between its fully distributed connectionist representations of symbols and the symbols themselves. That correspondence, however, will not be merely spatial as with semi-distributed representations, but functional and possibly non-isomorphic. As before, a description of the behavior of the system in terms of symbol structures might be possible, but a precise description of how the processing mechanism is implemented will only be possible at the level of individual units and activities.

In this thesis, the problem of systematicity and generativity will be studied using both semi-distributed and fully distributed representations.

1.9 Summary

As a conclusion to this chapter, we present below, in a concise form, the axioms and the main hypothesis of this thesis.
We believe that:

- Representations used in human cognitive processing of highly combinatorial domains display:

  P1 Compositionality

  P2 Systematicity

- Human cognitive processing involving representations of objects of highly combinatorial domains displays:

  P3 Inferential systematicity

  P4 Generativity

- Human cognitive learning in highly combinatorial domains displays:

  P5 Low interference

  P6 Rapid learning

- Some connectionist representational schemes are capable of displaying **P1** and **P2**.

The following four hypotheses are proposed:

H1 In combinatorial domains, **P3** will emerge with connectionist representations schemes displaying **P1** and **P2**

H2 In combinatorial domains, **P4** will emerge with connectionist representations schemes displaying **P1** and **P2**

H3 In combinatorial domains, **P5** will emerge with connectionist representations schemes displaying **P1** and **P2**

H4 In combinatorial domains, **P6** will emerge with connectionist representations schemes displaying **P1** and **P2**

Table 1.1: Summary of central claims and hypotheses of this thesis

	Strongly combinatorial domains		Weakly Combinatorial domains	
	Symbolic	Connectionist	Symbolic	Connectionist
Representations are:	Syntactic	Vectorial	Unaccounted	Vectorial
Compositional (**P1**)	Yes	Yes	Unaccounted	No
Systematic (**P2**)	Yes	Yes	Unaccounted	No
Processing is:	Syntactic	State changes	Unaccounted	States changes
Systematic (**P3**)	Yes	Yes* (**H1**)	Unaccounted	No
Generative (**P4**)	Yes	Yes* (**H2**)	Unaccounted	No
Learning is:	Syntactic	Weight changes	Unaccounted	Weight changes
Low interference (**P5**)	possibly yes	Yes* (**H3**)	Unaccounted	No
Fast learning (**P6**)	possibly yes	Yes* (**H4**)	Unaccounted	No

We believe that hypotheses **H1** through **H4** can be instantiated with the passage from context independence to content dependence, through learning, as a cognitive map would be formed.

Table 1.1 summarizes the connectionist claims and hypotheses stated above, and compares them with what we believe could be their symbolic counterpart. The stars correspond to hypotheses.

The label "Strongly combinatorial domains" refers to those domains that can (although approximately and incompletely sometimes) be described in terms of conceptual entities. The label "Weakly combinatorial domains", on the other hand, refers to domains for which conceptual descriptions are not possible, like visual fields and auditory signals. Since symbolic theories do not apply to these domains, representations, processing and learning are not accounted for by these theories. While symbolic learning theories are diverse, few do not allow, in principle, for fast learning with low interference. Symbolic learning in strongly combinatorial domains was therefore laeled "possible".

Chapter 2

Approach

2.1 Methodology

As stated in the previous chapter, the goal of this thesis is to perform and analyze experiments designed to show that, through the use of connectionist compositional and systematic representations, generativity and systematic processing are properties that can be learned and displayed by a connectionist network. In order to design such experiments, we need to make the following choices:

- Choose a domain with objects that admit compositional and systematic descriptions.
- Choose a connectionist representational scheme preserving compositionality and systematicity of objects in the domain.
- Choose a processing task(s) which will fail if it is (they are) not systematically inferential and generative.
- Choose a connectionist learning procedure and a connectionist processing medium.
- Make sure that we are not implementing a language of thought.

The underlying philosophy which will guide our choices will be that of simplicity, for two reasons.

First, using well known and understood neural network machinery on a domain which is simply and obviously compositional and systematic will ease analyses of our experiments. Designing a complex task, on the other hand, would make the problem of finding why it worked or did not much harder, if not unsolvable.

Second, and most important, the simplest task will persuade us, in case of failure, that there is but very little hope that connectionism can solve our problem in general. Indeed, if we cannot solve the problem even in the simplest environment, probability of failure for any other will be very high. On the other hand, in case of success, a simple experiment will allow for a better generalization to more complex tasks in the space of architectures, learning and processing paradigms. The simpler the paradigm and architecture on which principles have been discovered, the easier the generalization to more complex architectures or paradigms.

With these considerations in mind, we now proceed to describe the choices made.

2.2 Choice of the domain

Combinatorial domains are domains whose elements present a compositional and systematic structure. In such domains, elements in the domain are constructed by combining smaller elements, and the correct processing of larger elements can be generalized from correct processing of smaller elements of which they are composed, taking due consideration of the means of composition. For our main experiments, we have chosen one of the simplest combinatorial domains possible: Cartesian products of sets. If X_i are sets, for $i = 1, \ldots, n$, then our combinatorial domain will be

$$X = X_1 \times X_2 \times \ldots \times X_n = \{(x_1, x_2, \ldots, x_n) \mid x_i \in X_i\}$$

X will be the domain consisting of sequences of n elements, the ith element in each sequence being some member of X_i. For simplicity, all X_i's will have the same number of elements in our experiments, A. This domain, in addition to being simple, has the advantage that the parameters define it, n and A, can easily be varied.

A number of more limited experiments will also be conducted with the domain X consisting of English words, in order to study whether our experiments can indeed generalize to a more complex domain. That domain, of course has a much more complex structure than the one described above, and rules of legal combination for its individual letters cannot be specified. A domain with strong cross-positional constraints, such as the union of sets $XX' \cup YY'$, will also be considered (section 4.4.2).

2.3 Choice of the representational scheme

We will choose a connectionist representational scheme that allows for compositionality and systematicity, namely the tensor product scheme, as defined in the first chapter. Each element in X will be coded by adding the tensor product representations of each element of X_i it is composed of. The tensor product representation of an element of X_i is formed by performing the tensor product (outer product) of the representations of its structural role (a vector representing the set X_i it belongs to) and its filler (representation of the element x_i).

Fillers $x_i \in X_i$ will be coded as random binary vectors. Their representation is thus distributed, since each dimension in the representation vectors may participate in the coding of several fillers. Role vectors will simply be, for the main experiments reported in chapter 4, vectors of null activities with the exception of the ith coordinate of X_i which will have activity 1. These representations will thus be semi-distributed (fillers have distributed representations, roles have local representations), as defined in (Smolensky, 1987b). They simply amount to a simple concatenation of the representations of the x_i's. In chapter 5 the important case of fully-distributed tensor product representations, where roles also have distributed representations, will also be considered.

Thus, in the case $n = 3$, and with semi-local representations as described above, if the random binary vectors of activities representing $x_1 \in X_1$, $x_2 \in X_2$, $x_3 \in X_3$, are (1, 0, 1, 1, 0), (0, 0, 1, 0, 0) and (1, 1, 1, 0, 0), respectively, then the vector of activities representing (X_1, X_2, X_3) will simply be:

$$(1,0,1,1,0,0,0,1,0,0,1,1,1,0,0) =$$

$$(1,0,1,1,0) \otimes (1,0,0) +$$

$$(0,0,1,0,0) \otimes (0,1,0) +$$

$$(1,1,1,0,0) \otimes (0,0,1)$$

where $\otimes$ denotes the tensor product operation.

It is important to note that in our connectionist experiments, each element of each set is represented in the network as some pattern of activities. It is these patterns, of course, that matter, and not whatever name we find convenient to give to the elements. In this thesis we will use letters as convenient labels for these elements. As a further convenience we will use the same set of labels for each set X_i in the Cartesian product, but again this is of no consequence. (In particular the network is not charged in any way with discovering that we like to use the same letter to label an element in X_1 and an element in X_2.) Thus if $n = 4$, a typical element of X could be written (A, B, A, C) or, more simply, as the string ABAC. We call the number of elements in X_i, A, the alphabet size.

2.4 Choice of the processing task: induction

2.4.1 Introduction

Our goal in designing our experiments, as was expressed earlier, is to study whether compositionality and systematicity of representations can be exploited by simple connectionist learning and processing media to allow for generativity and systematic inference. The processing task we have chosen is one of set **induction**: For a given combinatorial domain of the form discussed in section 2.2, we will present a network with a small subset of training patterns representing objects of that domain, and test how the network has induced the combinatorial structure of the domain by measuring its competence on the domain. (The term competence is here used to refer to a measure, to be defined, of how well the system performs the task.) A successful induction will show that the network learned to be systematically inferential by exploiting the compositionality and systematicity of the representations. Furthermore, successful successive inductions obtained by increasing the number of sets X_i composing X's (while keeping the size of the subset of training patterns for X constant) will show that connectionist learning can be generative, if the networks' competence grow exponentially as a function of the complexity of the domain they learn.

One simple task we could study is one where a connectionist system would learn to be a **recognizer** of a structured domain. Given only a small sample of instances of the domain, the system, unable because of the size of the sample to perform any kind of rote learning of all individual elements of the domain, (although it would have no problem to perform rote learning of all the elements of the training set) would have to induce (and therefore be sensitive to) the structure of the domain to correctly recognize whether a future untrained pattern belongs to the domain or not.

Another more complex task we could consider would be one where, in addition to recognizing the domain by discriminating between members and non members of the domain, a connectionist system would learn to reproduce each pattern representing a member of the domain. In that case the system would learn to be, in addition to a recognizer of the domain, an **auto-associator**. Repeating experiments with domains of increasing size and structural complexity would, also, allow us to test for the property of generativity, which would be detected if an explosive growth in the system's competence was observed as the size and structural complexity of the domains increase.

Still more complex, we could ask a connectionist system to learn to act as a **memory model.** In addition to auto-associative capabilities, the system would have to learn to display capabilities typically found in human memory, such as pattern completion (that is, the system would have to learn to reproduce correctly a member of the domain even when only a part of its representation is presented), or resistance to noise (the system would have to reproduce correctly a member of the domain even when a degraded version of its representation is given).

Another task, independent of the previous one yet an extension of the second one, would be to associate, according to a predetermined mapping sensitive to the structure of the domain, members of the structured domain to their mappings. In that case, the system would learn to act as a **pattern associator**. Structure sensitivity and generativity could be detected following the same approach used with the auto-associator.

Faced with this preliminary and certainly not exhaustive list of possible approaches, we decided to choose the second one, for our main experiments, as a good trade-off in terms of implementation and testing complexity.[1] While the auto-associator task as mentioned above can only be successful if the processing mechanism involved is structure-sensitive and does allow testing for generativity, (this last property would be hard if not impossible to test with a recognizer) it does not require the kind of stringent restrictions, in terms of psychological modeling adequacy, that the design of a memory model would impose. It is, furthermore, easier to test. Indeed, testing for an adequate pattern completion capability alone would involve many tests, since there are many ways to present a partial representation to a system.

It is worthwhile to note that at a conceptual level, an auto-associator is a simple case of a pattern associator where the mapping relating input patterns of the domain and their output is the identity.[2] While presenting the advantage of being simpler, there is no reason to believe that results obtained with it could not be generalized to the case of a pattern associator.

With this preliminary choice of the auto-associator in mind, we now proceed to describe the processing task more formally. If $E = \{0,1\}^\nu$ is the space of all binary vectors of dimension ν containing all vectors representing members of a given domain X, $\|\ \|$ a norm on R^ν, and ϵ a strictly positive real number, then a network, given a small set of training examples of X that could learn a mapping f of the form:

[1] Small experiments testing the memory capabilities of some of our networks will be reported in chapter 4, however. In chapter 5, we will also report on experiments modeling the simpler recognition problem.

[2] Of course, since the auto-associator is asked to discriminate between members and non members of the domain, the mapping it has to implement, defined on all possible patterns rather than just the ones of the domain, is not the identity. This issue will be expanded shortly.

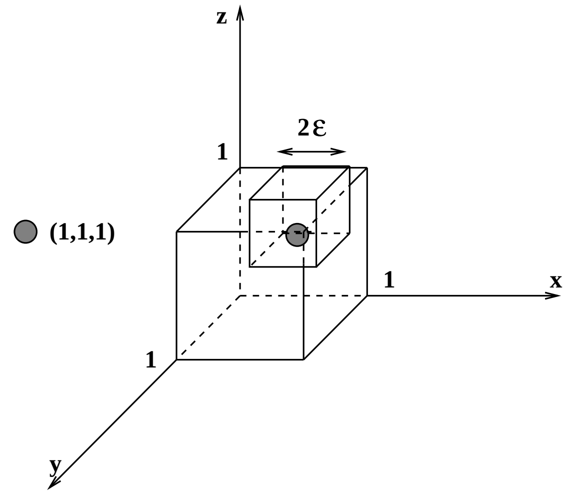


Figure 2.1: Auto-association with the L_∞ norm. If $\nu = 3$, the pattern (1,1,1) is allowed to be auto-associated with any point within the smaller cube, of side size 2ϵ, centered on the point (1,1,1).

$$f: \begin{array}{l} E \rightarrow R^\nu \\ x \mapsto f(x) \end{array} \quad | \quad \begin{cases} \|f(x) - x\| < \epsilon \text{ if } x \in X \\ \|f(x) - x\| > \epsilon \text{ if } x \notin X \end{cases} \tag{2.1}$$

would have induced the structure of the domain, interpreting a pattern z of E as recognized if and only if $\|f(z) - z\| < \epsilon$.

Note that we did not say anything about ϵ, which measures both the maximum distance allowed between any representations of members of the domain and their network associated vector, and the minimum distance between any vector which is not a representation of a member of the domain and its associated vector. Any chosen value would guarantee discrimination and thus learning of the structure of the domain. If, in addition, $\epsilon \in R$ is chosen such that

$$\forall (x_1, x_2) \in E^2, \|x_1 - x_2\| > 2\epsilon \tag{2.2}$$

then the network will discriminate between individual members of X and will therefore, [3] ,in the case of a successful learning of a function f, act not only as a recognizer but also as an auto-associator for members of X, associating each element $x \in E$ to $\tilde{X} \in E$, the closest element to $f(x)$.

Unless indicated otherwise, the norm we will use is the "infinity" norm $\| \ \|_\infty$ defined as $\|x\|_\infty = \sup_{i=1,..\nu} \|x_i\|$ where $x \in R^\nu$ has coordinates x_i, and ϵ will have a value of 0.4. This means, in practice, that we ask our system to auto-associate each representation of a member of X with an error of at most 0.4 along each dimension. In other words, we ask that all activities be "on the right side" of 0.5, and so by at least a distance of $0.5 - 0.4 = 0.1$.

From a geometrical perspective, the choice of the "infinity norm" can be interpreted in the following way. With the semi-distributed tensor product representation defined in section 2.3, members of X can be interpreted as being at certain corners of the ν dimensional hypercube defined by E, which has a corner at the origin and sides parallel to the axis and of length 1. The constraints on f state that the network associating a pattern with itself can do so with a maximum distance error (in terms of $\| \ \|_\infty$) of ϵ, that is that $f(x)$ is

[3] Let $X \in E$, $\|f(x) - x\| < \epsilon$, and let $\tilde{x}$ be the closest point in E to $f(X)$. Let us suppose that $\tilde{x} \neq x$. Then the inequalities $\|f(x) - \tilde{x}\| < \|f(x) - x\| < \epsilon$ holds. It follows that $\|x - \tilde{x}\| = \|x - f(x) + f(x) - \tilde{x}\| \leq \|x - f(x)\| + \|f(x) - \tilde{x}\| \leq \epsilon + \epsilon = 2\epsilon$, which contradicts condition 2.2. So $x = \tilde{x}$.

constrained to lie within the n-dimensional hypercube of sides of size 2ϵ centered on x. Figure 2.1 shows the two hypercubes in the case $\nu = 3$, for a pattern with coordinates (1,1,1). The use of the infinity norm can be contrasted from the use of an Euclidian norm, where an auto-associated output pattern would be constrained to lie in a hypersphere of radius ϵ centered on the input pattern, instead of the hypercube defined above.

The use of the infinity norm is valuable, practically, for the following reason. Although there is an uncountably infinite number of functions f that could satisfy equations 2.1 and 2.2, there is a finite number of binary function f_B's that satisfy equations 2.1 and 2.2.

The functions f_B's are simply the binary functions that would be implemented by a network computing any f satisfying the two constraints named above, where each output unit would be replaced by a binary threshold unit. The replacement would have the effect of thresholding activations smaller than ϵ to 0, and activations larger than $1 - \epsilon$ to 1. As we will see later, it is on the simple functions f_B's that our measures of both generalization power and discrimination will be based.

It is important, at this point, to reflect on just how a connectionist network could learn, or induce, a mapping f as defined by equation 2.1, and thus learn to be sensitive to the structure of the domain X.

The network, at the beginning, has only knowledge of individual bits and does not "know" that each pattern of X is made out of sub-patterns X_i's. During learning, as the connectionist learning algorithm chosen modifies the weights to ensure proper auto-association of the patterns of the sample training set, the network could learn to simply copy the bits of the input representation to the bits of the output representation. But what we will show is that, with certain architectural constraints, the learning process can avoid this simple mapping (which would prevent any discrimination between members and non-members the domain) by, instead, taking into consideration the structure of the domain, and "extracting", from the statistical regularities seen in the training sample, the structure of the domain.

We then hope that the network will behave in the same way on unseen patterns as it has learned to behave on trained ones. In other words, we hope that the network's behavior will **generalize** its learned behavior to unseen patterns. The process of generalization is a complex one, and worth addressing here.

2.4.2 Generalization

Generalization is one of the most pervasive inductive activities found in human reasoning. Indeed, as Holland et al. (1986) have written,

> Since Aristotle, generalization has been the paradigmatic form of inductive inference (page 231).

What, exactly, do we mean when we say we generalize? This complex question has been addressed both from a philosophical point of view, starting back from Aristotle, Hume and Popper to, for instance, (Thagard and Nisbett, 1982), (Holland *et al.*, 1986), and a psychological point of view (e.g.(Medin and Smith, 1984), (Holland *et al.*, 1986)). The problem has also, recently, been investigated by researchers in traditional artificial intelligence (Mitchell, 1982) (Michalski, 1983), theoretical learning (Valiant, 1984) (Blumer *et al.*, 1987), and neural networks (Denker *et al.*, 1987) (Samalam and Schwartz, 1989) (Schwartz *et al.*, 1990).

Mitchell (1982), in some pioneering work in the context of symbolic machine learning, formalized the problem in the following way. (The description below was heavily inspired by a review paper by Haussler (1987).)

For a given problem, the space Ω of all possible examples (including the subset of examples used for learning) is defined as the instance space Ω. Concepts of Ω are defined as arbitrary subsets of Ω, on which a space H of all concepts that can be described by a given concept description language (a generalization language) can be defined. Such a space H is called a hypothesis space.

Given a type of concept description language, we are interested, when solving a generalization problem, in finding an algorithm which would solve the problem for **any** hypothesis space H associated with **any** concept description language of that type. If $\mathbf{H}$ is a class of spaces H associated with concept description languages of a given type, then, a recognition algorithm for $\mathbf{H}$ is defined as an algorithm, defined on the set of all descriptions of instance spaces Ω and all samples (subsets) of an unknown concept in Ω, that either produces a hypothesis space H_Ω in $\mathbf{H}$ defined on Ω consistent with the sample or indicates that no such hypothesis exists.

Mitchell gives an example of such a recognition algorithm. He defines a version space, which we can denote $S_{H,Q}$, with respect to a sample Q of examples and a hypothesis space $H \in \mathbf{H}$ as the set of all hypotheses in H that are consistent with Q. The version space recognition algorithm, then, is one that, given a description of an instance space Ω and a sample Q of examples, examines each example in Q and shrinks the version space

$S_{H_\Omega,Q}$ associated with H_Ω defined with respect to Ω, by rejecting hypotheses that are not consistent with the example. When all examples in Q have been examined, the algorithm either returns that no hypothesis H_Ω has been found (when the version space is shrunk to the empty set), or returns a hypothesis of the shrunk version space. [4]

Note that there is no sense, within this framework, of a topology which could help us measure how "close" we are, when applying the recognition algorithm, to the target concept. We would need such a measure to formalize the idea that a true generalization algorithm is one that "closes in" or converges on the target concept. Valiant, in his "theory of the learnable" (1984), introduced and formalized such a concept of convergence within this framework.

He defines a learning algorithm as a recognition algorithm for **H** such that:

> For any instance space Ω,
> For any probability distribution on Ω and target concept in H_Ω,
> $\forall\epsilon > 0$, $\forall\delta < 1$,
> there exists a sample size such that the algorithm produces,
> with probability at least $1-\delta$,
> a hypothesis with error at most ϵ

Valiant also introduced the concept of computability, in the theoretical computer science sense, by considering the rate of convergence and the amount of computation required to produce a hypothesis. If the size of a concept c in H, $size(c)$, is defined as the number of symbols in the smallest description of c in the concept description language associated with H ($size(c)$ gives an approximate measure of the complexity of the target concept), then **H** is said to be polynomially learnable (in the sense of Valiant) if the sample size in the algorithm described above grows polynomially in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, $size(c)$ and the complexity parameters of **H**, and the algorithm runs in polynomial time with respect to the size of the sample.

2.4.3 Generalization and neural networks

Generalization ability prediction for a given problem and training set

The view of generalization as the process which consists of searching (inferring), when presented with a subset of a set of examples, a hypothesis among a set of alternative hypotheses which will be consistent with both the seen and unseen examples of the entire set, as formalized by Mitchell, and the refinement of such a view by Valiant who introduces the idea of convergence of hypotheses, then, allows for a clear understanding of the process of generalization which can take place within the connectionist learning framework. Figure 2.2 simply illustrates that view. Two hypotheses G1 and G2, in the space of all hypotheses consistent with the training set, are shown. Both could be the valid output of a recognition algorithm (in the sense of Mitchell). Only G2, however, consistent with the testing set, could be, on average, the output of a learning algorithm (in the sense of Valiant) because convergence restrictions in the algorithm would constrain it to output a hypothesis which would agree, with a high probability, with the testing set. Furthermore, a learning algorithm, in the sense of Valiant, would find, with a high probability, the hypothesis G2 independently of the specific training set. G2 would still, therefore, be found in the configuration illustrated in figure 2.3.

The view of the generalization process as expressed above and as formally defined by Valiant has recently been exploited (or independently reformulated), by a number of researchers in the neural network research community (e.g (Denker *et al.*, 1987) (Samalam and Schwartz, 1989) (Schwartz *et al.*, 1990)) and has led to important results.

Schwartz et al. (1990), in particular, have derived the following. Consider a network, defined by a set of weights W, with which we would like to learn a particular binary function $\bar{f}$ defined on $E = \{0,1\}^\nu$ and

[4] Mitchell's algorithm does more than that. In the context of symbolic concept description language, it takes into consideration a natural partial order of increasing generality defined on hypotheses of the version space, which is simply the partial order of set containment among hypothesizes. The algorithm can thus, if the version space has not been shrinked to the empty set, return a "maximally general" or "minimally general" hypothesis of the version space.

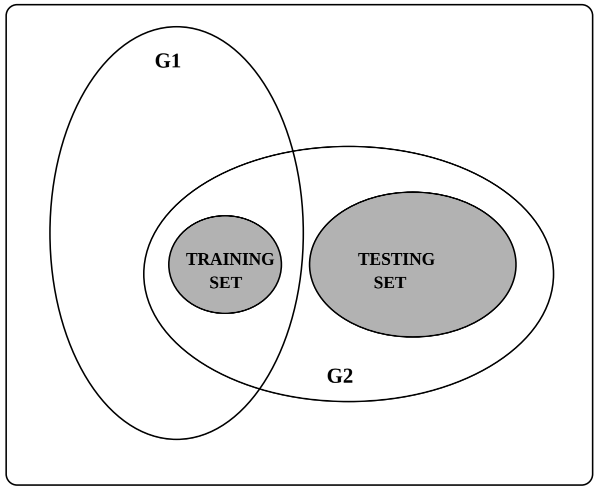


Figure 2.2: The space of generalizations

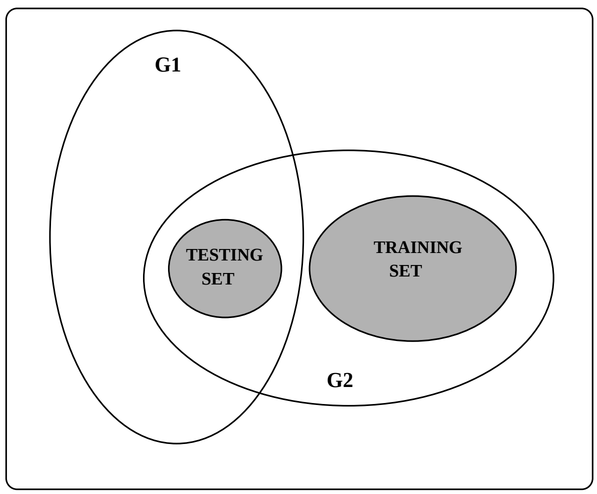


Figure 2.3: The space of generalizations with a different training set

mapping E to a single binary output. [5] The network is trained with a set of m input/output examples of $\bar{f}$, and is expected to generalize to $\bar{f}$. For each different configuration of weights W the network computes a particular function f. As the network learns from the training examples, its weights are modified and the network computes a new function f. The generalization ability of a function f computed by a given network can be defined as:

$$g(f) \; = \; \frac{2^{\nu} \; - \; d_H(f, \bar{f})}{2^{\nu}}$$

$g(f)$ measures the fraction of the number of input patterns on which f and $\bar{f}$ differ. (d_H is the Hamming distance between f and $\bar{f}$, or the number of bits different on their truth table.)

[5]The results below also apply to a network learning a binary function with multiple binary outputs, such as the functions f_B's with which we have been concerned. In that case, the term d_H in the definition of the generalization ability would be the Hamming distance between $\bar{f}$ and f_B, and the term 2^{ν} in that definition would have to be replaced by $\nu 2^{\nu}$.

If $\rho_0(g)$ is the probability density of generating a network prior to learning (generating random weights), with generalization ability g, then Schwartz et al. have shown that the average generalization ability G_m after m training examples is:

$$G_m = \frac{\int_0^1 g^{m+1}\rho_0(g)dg}{\int_0^1 g^m \rho_0(g)dg} \tag{2.3}$$

This derivation is obtained from probabilistic considerations, and rests on the following important assumption:

(1) The probability that the function f, computed by the network after $m-1$ examples, will correctly map the next input/output example m is, on average, $g(f)$.

Equality 2.3 holds for any given network architecture and learning technique, as long as assumption (1) described above is justified, and states that the generalization ability can, in theory, be computed **before** any learning. Although it can seldom be used in practice, because the space to explore to estimate $\rho_0(g)$ is too large, its existence has an important repercussion: Equation 2.3 means that if a particular average generalization ability G_m is observed, for a given network architecture, learning algorithm, processing task and training set of m examples, then any generalization ability for the same task, same training set and same network, but with a different learning algorithm, as long as assumption (1) is valid for that algorithm, will be G_m, on average. This is because $\rho_0(g)$ depends only on the function $\bar{f}$, the network architecture it is computed on and the testing set, but not on the particular learning algorithm used, as long as that learning algorithm can justify assumption (1).

Which learning algorithms verify assumption (1) is an open and empirical question. Schwartz et al. have shown, however, that the particular back-propagation learning algorithm ((Rumelhart *et al.*, 1986a), also, see section 2.6.1), on a small problem, the contiguity problem, does behave according to equation 2.3. (The contiguity problem, which consists in counting the number of blocks of contiguous 1's in the input pattern, was examined for patterns with low dimensionality). They have furthermore shown that the overall shape of the average generalization error $1-G_m$ when m is large enough can be predicted from equation 2.3, and is either a decreasing exponential or follows a decreasing power law, depending, respectively, on whether there is a finite gap in the distribution density $\rho_0(g)$ between g=1 and the next highest value of g for which $\rho_0(g)$ is non zero. These results are consistent with empirical results obtained by Ahmad (1988), who studied generalization ability of networks learning with the back-propagation algorithm the majority function (that function returns a 1 if the majority of binary inputs are 1's, 0 otherwise).

Although it remains to be proved, it is our intuition that assumption (1) will be satisfied for any learning algorithm based on a reduction of error, such as the back-propagation algorithm. There is a good chance, therefore, that our results will generalize well to other learning algorithms and will not crucially depend on the particular ones used in our experiments.

Generalization ability prediction for a given problem and any training set.

We stated earlier that the theoretical framework introduced by Valiant has had the merit of formalizing views of the generalization process. It has had, in addition, the advantage of allowing, in certain cases, proofs showing that certain problems were learnable as well as derivations of theoretical bounds on the number of examples needed to learn specific problems. Haussler (1986), for instance, has shown that a modified version of Mitchell's version space algorithm, the "One-sided algorithm for pure conjunctive concepts", which can be applied to version spaces described in terms of conjunctive features, is a learning algorithm in the sense of Valiant. In a related manner but within the neural network paradigm, Baum and Haussler (1989) have derived a theoretical upper bound on the number of training examples with which a neural network would need to be presented, to guarantee learning (in the sense of Valiant) of a specific task. It is important to note that whereas the results of Schwartz et al. concern estimates of the average generalization capability for a given network and a given training set, the theoretical upper bound computed by Baum and Haussler holds for any networks of a given architecture, and any given training set drawn randomly from the problem. Baum and Haussler's result originate from a result in statistics obtained by Vapnik and Chervonenkis (1971) who have shown the following:

$$P(\max_{\{all f\}} ||g_m(f) - g(f)|| > \epsilon) \; \leq \; 4F(2m)e^{\frac{\epsilon^2 m}{8}} \tag{2.4}$$

where F is a function that measures the maximum number of different binary functions, or the maximum number of dichotomies, that could be implemented by the network on any set of m training examples, and the left hand term measures the probability that the worst case estimation error exceeds ϵ. Vapnik and Chervonenkis have shown that the function f, called the growth function, is always either equal to 2^m or "slows down", (more precisely, is bounded by $m^{d_{VC}} + 1$) when m exceeds d_{VC}.

d_{VC} is called the Vapnik-Chervonenkis, or VC, dimension, which is infinite when $F(m)$ is equal to 2^m for all m. Baum and Haussler derived an upper bound on the VC dimension for a feed-forward network of linear threshold units with W weights and ν threshold units, by estimating the maximum number of dichotomies implementable by such a network, and from that proved that if $O(\frac{W}{\epsilon}\log\frac{\nu}{\epsilon})$ examples were used for training such a network which would correctly perform on a fraction $1 - \frac{\epsilon}{2}$ of this training set, then the network was guaranteed to perform correctly with probability approaching 1 on a fraction $1 - \epsilon$ of any other training set based on the same distribution, when $\epsilon \; \leq \; \frac{1}{8}$.

A feed-forward network with 30 input and output units, and 20 hidden units, then, would need, to be guaranteed to correctly perform on 7/8 of any testing set, ($\epsilon = 1/8$)to correctly classify 15/16 of a randomly chosen training set of size

$$\frac{W}{\epsilon}\log\frac{\nu}{\epsilon} = \frac{2\times 30\times 20}{\frac{1}{8}}\log\frac{20+30}{\frac{1}{8}} \;=\; 82,944$$

In practice, however, much less sampling might be required, depending on the training task.

Sontag (1990) has shown that such an upper-bound could be further reduced when non-linear sigmoidal activations functions are are used.

With respect to probabilistic generalization ability analysis, where predictions are estimated for any randomly chosen training set for a given problem, we should also mention approaches to the problem that have originated from the field of statistical mechanics. When viewed as collections of simple interacting units, neural networks closely resemble large-scale atomic physical systems, and their dynamics can, indeed, be studied in a similar way (Tishby *et al.*, 1989). Drawing on (Gardner and Derrida, 1989) who used mean field analysis to study optimal storage capacity of Hopfield networks, a calculation of the generalization ability as a function of the size of the randomly chosen training set, this independently of the learning algorithm used, was possible (Opper *et al.*, 1990). In (Krogh and Hertz, To appear), a related approach has led to the theoretical estimation of the generalization ability of linear perceptrons.

Mathematical approach to the problem of generalization

Wolpert (1990b) (1990c) has approached the problem of generalization from a highly abstract mathematical standpoint. Defining generalizers as a countably infinite set of functions, with certain properties such as input and output space translation, rotation, parity inversion, and scaling invariance, Wolpert spends considerable time narrowing his definition of generalizers to allow for properties deemed valuable in a generalizer, such as, for example, convergence (in a sense related to Valiant's). The theory, however, is as of now incomplete, and the title of (Wolpert, 1990a), "Constructing a generalizer superior to NETtalk via a mathematical theory of generalization", might be misleading. In this paper, it is shown that a particular processor, one introduced by (Stanfill and Waltz, 1986), does better in terms of generalization capability, while qualifying as a generalizer in the sense of Wolpert, than the "reading-aloud" connectionist network NETtalk (Sejnowski and Rosenberg, 1987), for the same task.

2.4.4 Empirical studies

Apart from experimental studies confirming theoretical results similar to those obtained by Schwartz et al. and reported in (Denker *et al.*, 1987) (Schwartz *et al.*, 1990) (Samalam and Schwartz, 1989), (Tishby *et al.*, 1989), little empirical research has been done on generalization.

Ahmad (1988), as mentioned earlier, has studied how the perceptron learns the majority function and has derived a number of results. He has shown that in his experiments, the fraction of misclassified instances decreases exponentially with the size of the training sets. He has also shown that the size of the training set needed to obtain a fixed competence level scales linearly with the number of input units, and that the nature of representations used play a crucial role in generalization performance.

Other experimental research activities on generalization have been targeted at improving generalization (e.g (Mozer and Smolensky, 1989) (Chauvin, 1990) (Weigend *et al.*, 1990) (Yu and Simmons, 1990)) by modifying learning algorithms. We will briefly discuss these in chapter 4.

2.4.5 Connectionist modeling and generalization

Within the connectionist cognitive modeling paradigm, generalization has, surprisingly, received little attention as a property which would be worth studying for its own sake, in spite of the inherent aptitude at generalization that neural networks seem to possess. As was briefly mentioned in chapter 1, generalization has until now been a property of connectionist networks which, in the specific context of cognitive modeling, has been merely observed. Studies involving generalization have invariably followed the same pattern: First a selection of input/output examples from the task at hand is chosen. Then a network is trained on a subset of that selection. It is then tested on the small number of remaining examples (10-20%) of the selected set. Generalizations are few, and do not outnumber training examples. Current models suggests that in order to obtain correct performance on a target set of inputs, a network needs to be trained on a sizable fraction (between 25% and 75%) of the learning set. This state of affairs, no doubt, leaves little room for the view that connectionist systems could perhaps exhibit massive generalization, a much needed property if one believes that connectionism is not in theory incompatible with cognitive properties such as generativity or systematic inference.

The question of how many possible generalizations could occur, in the context of a cognitive modeling task, has never, to our knowledge, been investigated. Our results, as we will see, suggest that much more generalization can be obtained than has previously been reported. This is in part due to the fact that our experiments will be designed to look specifically for *all* possible generalized patterns. It is also because our experiments will deal with *combinatorial* domains, where patterns of the training set follow combinatorial and systematic principles allowing for generativity and systematic inference, and thus large number of generalizations.

2.5 Virtual generalization

2.5.1 Introduction

Looking at how well a network has generalized from training examples, for a given task, by counting how many generalizations are obtained is one measure of the network's competence, but a crude one since it does not give any indication on how well the network would converge, given more training examples, on the target mapping defined by the task. One way of measuring such convergence, as presented earlier, would be to test the network on how it "closes in", as size of training sets increase, on the target mapping, by testing how generalization ability of the network increases with more examples. Another similar way of measuring convergence, and one which can be more directly related to human learning, would be to measure both how easily the network, once it has learned a training set, can learn an unseen example of the domain at hand, and how much that unseen example would interfere in the performance of previously learned items.

Examples of such easy and interference-free learning abound in the human learning paradigm. In the context of linguistic competence, for instance, sentences that we have never heard before and which contain new information that we wish to learn can be, with very fast and limited learning, understood and memorized easily, with little, if any, interference with previous linguistic knowledge. In general, any human learning task involving examples with underlying regularities requires, once a level of competence is acquired, (that is, once the regularities have been discovered), little learning for additional examples. These examples, furthermore, cause little disruption to the acquired competence, because of the very fact that they share in the regularities already discovered.

Within the connectionist modeling paradigm, then, a network generalizing poorly would learn an unseen example laboriously and that example, furthermore, would disrupt performance of examples of the previously learned training set. On the contrary, one would expect, in a network which is generalizing well in terms of convergence, that an unseen example of the domain would be learned easily, and cause no interference with the correct performance of examples of the original training set. This view has led us to define such an unseen example as a **virtual generalization** [6] Simply stated, then, a virtual generalization is an example which, although not correctly generalized, can be learned quickly while leaving performance on the previously learned examples intact.

By leaving performance of the previously learned examples intact, we simply mean that each example of the original training set, with respect to the criterion for learning used for members of that set, performs to criterion after the virtual generalization is learned. By quick learning, we mean that, in connectionist terms, the number of presentations needed to train a virtual generalization is small compared to the number of presentations which was needed to train the original set. Our original intuition, based on the observation that interference and quick learning are intrinsically related, was that any example which would cause no interference could be learned in just one presentation. Some empirical studies, performed with a specific learning algorithm, convinced us, however, that this is almost always the case, but not quite. In other words, there are examples, for some learning tasks, which can be learned without causing interference, but not in just one presentation. Let us say, then, and for now, that what me mean when we use the term "quick learning", is simply that the unseen example can, most probably, be learned in just one presentation, or can be learned, in the worse case, with a small fraction of the number of training presentations which were needed to train adequately the original set of examples.

A virtual generalization, then, is a novel input which can be trained, in a very small number of trials, to criterion, while leaving performance on the training set error-free. For computational time reasons, we restricted the number of learning trials when testing for a virtual generalization to 5. All our results concerning their number are thus lower bounds only: The use of lower learning rates and/or larger number of trials could yield higher numbers. We will henceforth mean virtual generalizations that can be learned in 5 trials or less when we refer to virtual generalizations.

2.5.2 Connectionist modeling and interference

Until recently, connectionist relearning of new items has followed a typical scenario, [7] which can be summarized by the following account: First the weights of the network are given small random values, and the learning procedure chosen is applied a large number of times until the patterns belonging to the original set are learned. Then, if a new pattern needs to be learned, it is simply added to the set of original patterns and the network is retrained again *from scratch*, all the weights being reinitialized again, the learning procedure being again applied as before thousands of times. New items to be learned, then, were in current practice intermingled with all the previously trained inputs and subjected again to the lengthy and rather laborious training algorithm.

Research by McCloskey and Cohen (1989) suggested that this state of affairs was unavoidable. In a first set of experiments, the authors used a standard three-layer back-propagation network to model sequential learning of simple arithmetic facts. Single digits were coded using distributed representations, while the representation of a two digit number was the concatenation of the two single-digit representations. Four units were used to code either the "+" sign of addition or the " $\times$" of multiplication. When the network was first trained on all 200 single-digit addition and multiplication problems, it performed very well. However, when it was trained on only the "1's" addition problem (that is, 1+1 through 1+ 9 and 2+1 through 9+1), and then, subsequently, on the "2's" addition problem, then it was found that training on the "2's" addition problem interfered with previous knowledge on the "1's" addition problem in such a way that the network performed very poorly on the the "1's" addition problem. In other words, the network, successfully trained on the first set of items, "unlearned" that first set when it was trained on the second one.

[6] In earlier publications, ((Brousse and Smolensky, 1989b), (Brousse and Smolensky, 1989a), (Brousse and Smolensky, 1990)), we used the term "virtual memory", instead. The term "virtual generalization", which obviously refers to learning tasks where there is an underlying mapping or structure to generalize on, and which in addition does not require that the connectionist network studied have properties of a memory model, is used here.

[7] An exception should be made concerning this, in the case of category learning modeled by adaptive resonance theory (Grossberg, 1976) (Carpenter and Grossberg, 1987). Within the framework of competitive learning, the adaptive resonance theory scheme addresses the problem of classifying new patterns into new or previously discovered categories by forming new ones as needed, depending on how well the current pattern fits within a previous category. This unsupervised "one-trial" learning scheme can thus classify new patterns in a continuous manner.

In another set of experiments, McCloskey and Cohen modeled a classic experiment involving human learning retroactive interference (Barnes and Underwood, 1959). In that experiment, subjects are asked to first learn a list of 8 paired-associates of the form A-B, and then to learn a second list of the form A-C. The cue in the second list is thus the same as in the first list. It is shown in that experiment that human retroactive interference occurs, as performance on the initial list is disrupted by learning of the second list. When modeling the task with standard three-layer back-propagation networks, McCloskey and Cohen found that the networks, too, displayed interference. But this interference was "catastrophic" in that it was much higher than in the case of the human experiment. While subjects still retained some competence on the first set of paired associates after having learned the second one, the connectionist networks unlearned dramatically the first sets when they were trained on the second one. No architecture or parameter adjusting, furthermore, allowed McCloskey and Cohen to obtain, with their networks, interference results close to the limited interference observed in the psychological experiment.

From a related perspective, Ratcliff (1990) studied the feasibility of modeling learning and forgetting in recognition memory, with multi-layer back-propagation encoder (auto-associator) networks. In a first set of experiments, a network was trained on three items represented by orthogonal vectors of length four, and a fourth one was then trained separately on top of the three previously learned items. Testing on all items showed that the training of the last item severely disrupted the recognition of the previous items. Variants of the experiment, made by adding more hidden units, or allowing only modification of small weights during learning of the last item, or by adding new hidden units and allowing only modification for weights connected to the new units when training on the last item, yielded the conclusion that it was not possible to have both the first three items and the last one be correctly reproduced, since the last item could only be reproduced correctly at the expense of the three previous ones, or vice-versa. Similar experiments were repeated using larger networks (and thus larger representations) and various presentation order schemes, and yielded similar conclusions.

In another set of experiments, Ratcliff investigated whether multi-layer back-propagation encoders could properly model recognition, where items need only be discriminated. Although the first set of experiments had clearly suggested that the networks used could not model a recognition **memory** (where each item needs to be properly recalled, that is, correctly reproduced) because of dramatic unlearning, it was found that discrimination measures performed on all the items yielded reasonable values consistent with human data. Studies of such discrimination measures as a function of amount of rehearsal, however, produced results inconsistent with human data. Not surprisingly, the kind of massive interference caused by learning of new items yielded a discrimination function which was either non-monotonic or decreasing with rehearsal amounts, a result inconsistent with human data where that function is invariably monotonically increasing. As before, several variations on the learning scheme and the architecture used did not significantly improve the ability of the networks at modeling the task.

A number of architectures or modified training algorithms/techniques have been devised to reduce the kind of massive interference reported by Ratcliff and McCloskey & Cohen. It should be mentioned, before we proceed to briefly present these, that connectionist interference is not, per se, a harmful property, since it is also present in certain human cognitive tasks. It is only a problem when it is catastrophic, that is, when its magnitude in a modeled task is much higher in a connectionist system than the one observed in the corresponding human processing experiment, as was the case with the paired-associates or recognition connectionist models.

The magnitude of interference observed in a connectionist model, relative to the magnitude observed in the corresponding human task, depends obviously on the realism of the model. Hetherington and Seidenberg, (1989), for example, replicated the McCloskey and Cohen experiments on the simple arithmetic facts problem, and showed that things were not as bad as they seemed. While learning the "2's" addition problem set of facts after learning the "1's" set of facts indeed caused "catastrophic interference" that seriously impaired performance on the first set, they showed that knowledge about the first set had not be completely erased, and little relearning on the first set was enough to induce good performance on both sets. If each set of " i's" addition facts, for i varying from 1 to 5, was learned serially but along with the "$i-1$'s" addition facts and "$i+1$'s" addition facts, in a serial "sliding window" manner, they showed, furthermore, that the whole addition problem could be learned without catastrophic interference. That more realistical learning scheme, then, which was a better approximation of the learning scheme applied by children when they learn arithmetic facts, effectively dealt with the problem of catastrophic interference in that case.

Interference depends, obviously also, on the learning technique and architecture used. Kortge, (1990), for example, replicated some of Ratcliff's experiments with a slightly modified learning technique (larger initial weights were used and activation values were constrained to be in a [-0.5, 0.5], instead of [0,1], range) and obtained much smaller interference. Relatedly, Estes (1991) mentions that replications of some of Ratcliff's experiments on different architectures, still however within the feed-forward back-propagation paradigm, yielded interference which was consistent, in magnitude, with corresponding interference found in humans.

It seems, then, that interference might not be as dramatic as a problem as earlier studies tended to show, in domains where human interference is significant. For processing tasks that involve very little human interference, however, the connectionist approach does not seem an easy one: Slight variations on typical learning techniques or architectures could reduce interference, but that would not be enough. Such processing tasks include episodic memorizing, where facts are only presented once but are quickly and easily remembered, with no interference, and processing in domains characterized by high regularity or structure -the kind of domains we are concerned with.

Episodic memory connectionist modeling, not surprisingly, has therefore made use of rather radical modifications of existing architectures and learning techniques.

Kortge, (1990), for instance, has devised a modification of the standard generalized delta rule, where weight modification to ensure proper processing of a new pattern takes into account the "novelty" of that new pattern with respect to old ones previously learned. In that case, then, weights are not modified according to the difference between output and target patterns, as with the standard delta rule, but are modified instead according to the "novelty pattern", where the novelty pattern is defined, heuristically, as:

- For the input layer: the difference between the output and target pattern.
- For the hidden layer: the output at the hidden layer of the difference vector defined above, fed at the input layer.

(The hidden novelty vector is also set to zero under certain circumstances). This modified learning rule has the net effect of changing less those weights coding well learned patterns, while still allowing for gradient (although not steepest) descent in total error. Experiments performed with such a modified learning rule yielded reduced interference.

Otwell (1990), in an ingenious scheme, uses recurrent novelty filters (Kohonen, 1984)[8] to allow incremental and interference-free back-propagation learning, by using the activities of a "filtered out" input pattern, instead of the activities of the input pattern, in the delta rule. Simulation results showed that interference-free incremental learning was possible, although a large number of hidden units (larger than the number of training patterns) were needed, which significantly reduced the present applicability of this approach.

French (1991), somewhat relatedly, has proposed an "activation sharpening" technique, where activations in the hidden layer are sharpened, that is, artificially increased where they are significant, and artificially decreased where they are small. Weights of the connections between the input and output layer are then modified according to the new association. The idea behind this scheme is to reduce activation overlap among different patterns, which causes interference. Simulation results showed that interference, again, was significantly reduced.

A final example of significant departure from standard architectures and learning techniques to reduce interference is seen in (Sloman and Rumelhart, To appear), where a Willshaw network (Willshaw, 1981) with additional "episodic" units, which act as gating units and context units, is used to model episodic memory. Interference is avoided as the extra connections involved with the episodic units develop their own representations when new learning occurs. Such representations prevents new knowledge from interfering with previous information stored in the other connections of the network.

[8]Recurrent novelty filters are a variant of recurrent auto-associative linear networks learning with the delta rule. It can be shown (Kohonen *et al.*, 1981) that such networks output, when presented with a pattern represented by a vector p, the vector $\tilde{p}$, where $\tilde{p}$ is the orthogonal projection of p onto the subspace spanned by the vectors representing the previously learned patterns. $p - \tilde{p}$ can thus be interpreted as the "filtered out" component of p which is "maximally new", with respect to the stored patterns. Novelty filters are networks which, presented with a pattern represented by the vector p, output only the "novel" component $p - \tilde{p}$.

2.5.3 Combinatorial domains and connectionist interference

These research results, then, would seem to indicate that there is but little hope that learning of new examples for our chosen task will yield little interference, unless we depart from traditional architectures and learning algorithms.

But for processing on highly structured domains, in fact, we hypothesize that connectionist interference will not be a serious problem.

McCloskey and Cohen gave a good explanation of why catastrophic interference can appear, which can be reworded and generalized as follows. When a learning algorithm based on iterative global error minimization, like the back-propagation learning scheme, is used to train a network on a set of patterns, the learning process can be viewed as a trajectory in weight space where each point along the trajectory further minimizes the total added error for all patterns of the training set (section 2.6.1). The back-propagation learning algorithm, then, can be viewed as a minimization procedure that performs a gradient descent in the multi-dimensional weight landscape where the total error is represented as a function of the weights.

When concurrent training is used, that is, when all patterns of the training set are learned simultaneously, all patterns are responsible for small steps in the trajectory towards global minimization, and thus equally influence the trajectory. If patterns are trained one at a time, on the other hand, the learning algorithm will first minimize the error of the first pattern, by following a trajectory in weight space minimizing that error, and will then move to a point that corresponds to a minimization of error of the second pattern. If these two points are close, or if the second point is also one that minimizes the first error, then learning of the second pattern will not have interfered with performance on the first pattern. In general, however, there is no reason to expect that the two end-points of the two trajectories associated with the separate and successive learning of two different patterns will be close or will both minimize the error for both patterns.

In structured domains, however, examples all share underlying regularities. Patterns representing members of the domain, therefore, are not random but all share common features. This is **precisely** why the network can generalize, in the first place. Initial learning of the training set will thus, it can be hypothesized, induce a structured landscape that allows untrained items of the domain to be learned with "small" independent learning trajectories. The new error landscapes corresponding to the total error of patterns of the training set plus the new pattern might not be significantly different and the small move in weight space due to the learning of the new pattern will not result in a significant altitude loss.

An alternative yet related hypothesis concerning interference-free learning in highly structured domains can also be formulated as follows: As was shown by (Hopfield, 1982) and (Smolensky, 1983), processing in a network can be interpreted as a move to a state of activity minimizing energy, or maximizing harmony. Learning consists in building, by changing connection strengths, the energy function according to which the training examples have low energy, or high harmony. As a result of this process, other states that are not part of the training set will typically become energy minima: these states determine the network's generalizations. When the domain at hand is a structured domain, we can hypothesize that the energy function built through training will put energy minima at the training examples by a process that simultaneously puts minima at a number of other combinations of the subpatterns whose recombinations form the training examples. (This will probably not happen if the domain is not structured and contains patterns not systematically related). Learning a new pattern of the domain, then, which does not yet constitute an energy minima but which is close, will involve, we hypothesize, minimal training in such a way that disruption of the existing distribution of energy minima (interference) will not occur.

2.6 Choice of the learning and processing machinery

The connectionist learning and processing technique we will use is a standard and much used one: auto-association in a feed-forward network using standard back-propagation learning (Werbos, 1974) (Le Cun, 1985) (Parker, 1985), (Rumelhart *et al.*, 1986a). A variant of back-propagation learning using the technique of weights elimination (Weigend *et al.*, 1991) is also used later on in this thesis (chapter 6). Below we describe the network used, to then present the reasons for our choices.

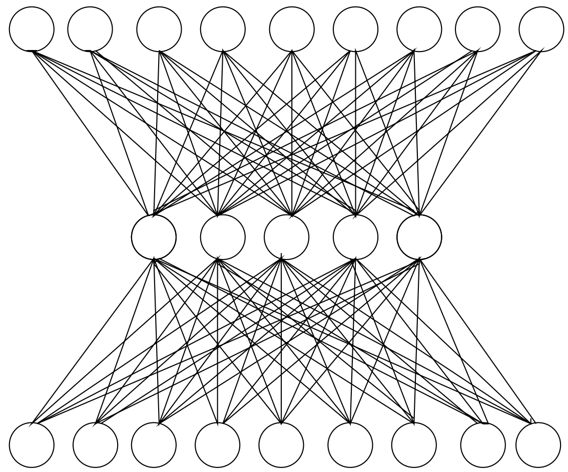


Figure 2.4: A three-layer back-propagation architecture

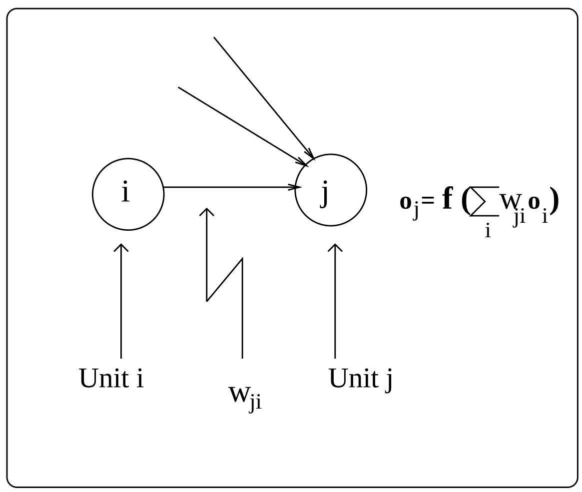


Figure 2.5: Units and weights

2.6.1 Description

The networks which will be used in our experiments are three-layer feed-forward networks (figure 2.4) in which the input layer and output layer have N units and the hidden layer has H units. Each training input (one of the elements of X) is represented as a pattern of activity on the N input units following the tensor product representational scheme described earlier. The target output on the N output units is identical to the input pattern: the network must therefore associate each element of X with itself. The network is trained with standard back-propagation, the units of the network being semi-linear units (figures 2.5 and 2.6), where each unit j which is not an input unit computes its activity $o_j = f(\sum_i W_{ji} o_i)$ by feeding to a semi-linear sigmoidal function, f, the sum of the weighted activities of the units i it is connected to. The function f used for our experiments was, unless indicated otherwise, the standard:

$$f : \begin{array}{l} R \rightarrow R \\ x \mapsto f(x) = \frac{1}{1 + e^{-x}} \end{array}$$

Learning is done with the generalized delta rule. The basic idea behind this learning procedure is the following: Within an epoch cycle, all input/output pattern pairs are presented to the network, and at each

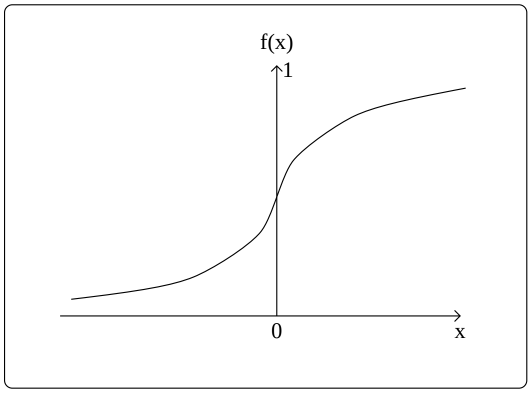


Figure 2.6: The semi-linear function used

presentation, an association between the specific input and output patterns presented is partially learned. The process is repeated over until all input/output pattern pairs are associated according to a specified error criterion. At each presentation, learning is done in two steps. In the first step, the input pattern is presented and propagated forward. The output pattern present in the last layer of units is then compared to the teaching input. The error (difference between actual value and target value) of each unit is then computed and "back-propagated" through the network, where weights changes are performed in order to reduce the error, according to the following generalized delta rule:

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi}$$

with

$$\delta_{pj} = \begin{cases} f'(i_{pj}) \sum_k \delta_k w_{kj} & \text{if unit } j \text{ is not an output unit} \\ f'(pj)\,(t_{pj} \; - \; o_{pj}) & \text{if unit } j \text{ is an output unit} \end{cases}$$

where i_{pj}, o_{pj}, t_{pj} are the input, output, and target activation values of the jth unit, respectively, when a pattern p is presented, $f'(i_{pj})$ is the derivative of the semilinear activation function which maps the total input (weighted sum of activities of units with incoming connections) to unit j, (figure 2.5) and η is the learning rate.

The generalized delta rule therefore provides a recursive top-down [9] formula for updating weights: Updates of weights connecting to the output layer are directly computed while weights of lower layers are successively updated as a function of updates of weights connecting to the layer above.

Let E_p be the error, in terms of least squares, between the output pattern and the target pattern.

$$E_p = \frac{1}{2} \sum_{j=1}^{N} (t_{pj} - o_{pj})^2$$

We show here, following (Rumelhart *et al.*, 1986a), that the back-propagation algorithm performs a stochastic approximation of a gradient descent in the total error over all patterns, $E = \sum_p E_p$. For this we will show that

$$\Delta_p w_{ji} = -\eta \, \frac{\partial E_p}{\partial w_{ji}}$$

[9] By convention, the top layer is the output layer and the bottom layer is the input later.

Since

$$\frac{\partial E}{\partial w_{ji}} = \sum_p \frac{\partial E_p}{\partial w_{ji}}$$

and since, in practice, values of η are small, we are insured that, although weights are changed after each pattern presentation, the net result is close to a true gradient descent in E where weights would only be changed after all patterns are presented.

Using the chain rule, we obtain,

$$-\eta \frac{\partial E_p}{\partial w_{ji}} = -\eta \frac{\partial E_p}{\partial i_{pj}} \frac{\partial i_{pj}}{\partial w_{ji}} = -\eta \frac{\partial E_p}{\partial i_{pj}} o_{pi}$$

Now since

$$\frac{\partial E_p}{\partial i_{pj}} = \frac{\partial o_{pj}}{\partial i_{pj}} \frac{\partial E_p}{\partial o_{pj}} = f'(i_{pj}) \frac{\partial E_p}{\partial o_{pj}}$$

and since

$$\frac{\partial E_p}{\partial o_{pj}} = \sum_k \frac{\partial E_p}{\partial i_{pk}} \frac{\partial i_{pk}}{\partial o_{pj}} = \sum_k w_{kj} \frac{\partial E_p}{\partial i_{pj}}$$

we have, with

$$\delta_j = -\frac{\partial E_p}{\partial i_{pj}}$$

$$-\eta \frac{\partial E_p}{\partial w_{ji}} = \eta o_{pi} f'(i_{pj}) \sum_k \delta_k w_{kj}$$

with, following from the definition of E_p,

$$\delta_j = f'(i_{pj}) \, (t_{pj} - o_{pj})$$

if j is the index of an output unit.

It must also be noticed that, as a gradient descent procedure, the generalized delta rule is bound by the problem of local minima. In practice, however, a large number of simulations have shown that local minima are rarely obtained. To allow for a maximal learning rate without leading to local minima oscillations, a momentum term α, increasing the effect of previous weight changes on the current descent direction, can be incorporated in the learning rule. The learning rule then becomes:

$$\Delta w_{ji}(t+1) = \eta(\delta_{pj} o_{pi}) + \alpha \Delta w_{ji}(t)$$

2.6.2 Rationale

Reasons for the back-propagation choice are numerous:

First back-propagation networks can be considered as one of the generic "pillars" of connectionist modeling and have been extensively used for cognitive modeling, in area as diverse as natural language processing (e.g (Hanson and Kegl, 1987), (Kukich, 1988)), speech pronunciation ((Sejnowski and Rosenberg, 1987)), control ((Miyata, 1988)), and expert systems, ((Fozzard *et al.*, 1989)) to cite only a few applications.

Second, their use is generally simple and straight-forward.

Third they belong to the class of models that self-organize: Their hidden layer(s) can develop internal representations which are used by the network to perform the designated task.

Finally, and most importantly, experimenting with the most straightforward and least special architecture will convince us, in case of success, that our results were not due to special architectural idiosyncrasies.

Chapter 3

Combinatorial domains

In the first chapter, we stated the central hypotheses of this thesis: that generativity and systematic processing could arise in combinatorial domains in which objects have compositional and systematic representations. In this chapter an attempt is made to formally define what will be meant by combinatorial domain, when the the laws of combination are concatenative.

The notion of combinatorial complexity, furthermore, is explored with such laws. Such a notion is central for any study of generativity in a growingly complex domain, which by definition involves explosive growth of competence as the complexity of the domain increases. The concept of "structure-preserving" connectionist representations, that is, representations which are most likely to faithfully represent the kind of combinatorial constraints within objects of a combinatorial domain, is furthermore explored. [1].

3.1 Combinatorial representations and combinatorial domains

3.1.1 Combinatorial representations

Intuitively, we say that a represented domain is combinatorial if it consists of combinations of representations of atomic elements. The set of written representations of grammatically well-formed English sentences, for instance, is combinatorial because it consists of all syntactically correct combinations of written representations of English words. Similarly, the set of all written English words is combinatorial, as it consists of orthographically correct combinations of representations of individual letters. When the representational scheme is symbolic, (as is the case with written language), combinations consists of the simple operation of concatenation. The representation of the word "GREEN", for instance, is made from the combination, or concatenation, of the letters "G", "R","E", and "N". Likewise, written well-formed sentences are made from the concatenation of English words and the "blank" symbol.

Intuitively, then, a represented domain X of objects is a combinatorial domain if it is represented by a mapping R such that the set of representations of the objects of X falls in a Cartesian product. More formally:

Definition 1 Let $X = \{x_1, \ldots, x_p\}$ be a domain of p elements, where each element can be described by at most b bits.
A **combinatorial representation** of X is an injective mapping

$$R: \begin{array}{ccl} X & \rightarrow & D \subset \Pi_{i=1}^{n} A_i \\ x & \mapsto & R(x) = (d_1^x, \ldots, d_n^x) \end{array}$$

(where $\forall i \in \{1, \ldots, n\}, d_i^x \in A_i$) and
(i) $1 < n < p$

[1] Concatenative laws of combination are but one example of combination laws Other examples include recursive laws, like gramatical laws, that recursively govern *combinations* of combinations of elements. An attempt to generalize the results reported in this chapter to such laws will not be made here, although a possible direction will be briefly discussed in section 3.3.2.

Condition **(i)** states that the representation of a domain X that simply consists of a listing of the individual representations of elements of X taken as a whole, that is, taken as atoms and not further decomposable, is not a combinatorial representation. If X is the set of the first 10 integers, for instance, the Arabic decimal representation (Adr) of X

$$R: \begin{array}{ccl} X & \rightarrow & \{0,1,2,3,4,5,6,7,8,9\} \\ x & \mapsto & R(x) = \mathrm{Adr}(x) \end{array}$$

is not a combinatorial representation, since condition **(i)** is not satisfied.

3.1.2 Examples

1. Let X be the set of all 4-letter strings of the alphabet $A = \{a, b, \ldots, z\}$, and let x_i be the ith letter of x. The written representation R_w of X

$$R_w: \begin{array}{ccl} X & \rightarrow & D = \Pi_{i=1}^{4} A = A^4 \\ x & \mapsto & R_w(x) = (x\_1, x\_2, x\_3, x\_4) \end{array}$$

is a combinatorial representation, with $1 < (n = 4) < (p = 26^4)$. In written language, $R_w(x)$ is simply written as $x\_1 x\_2 x\_3 x\_4$. For simplicity, this last notation will henceforth mostly be used.

2. Let $X = \{0, \ldots, p\}$ be the set of the first $p + 1$ integers. The binary representation R_b of X

$$R_b: \begin{array}{ccl} X & \rightarrow & D = \{0,1\}^P \\ x & \mapsto & R_b(x) = (x\_1, x\_2, \ldots, x\_P) \end{array}$$

where [2] $P = \lfloor \log_2(p)$ and x_i is the ith binary digit of the binary representation of P bits "left-padded" with 0's, of x, is a combinatorial representation.

3. Any domain $X = \{x_1, \ldots, x_p\} \subset Z$, where Z is the set of all positive or negative integers, has a combinatorial representation, for instance its binary representation:
Let $P = \lfloor \log_2(p)$. Then,

$$R_{bin}: \begin{array}{ccl} X & \rightarrow & D = \{+,-\} \times \{0,1\}^P \\ x_p & \mapsto & R_{bin}(p) = (\text{binary representations of p}) \end{array}$$

is a combinatorial representation, as $P + 1 < p$.

4. Let $X = \{1, \ldots, p\}$ be the set of the first p integers. The "thermometer" representation of X :

$$R_t: \begin{array}{ccl} X & \rightarrow & D = \Pi_{i=1}^{n}\{0,1\} \\ x & \mapsto & R_t(x) = (\underbrace{1,1,1,1}_{x}, 0, \ldots, 0) \end{array}$$

is not a combinatorial representation, since $n = p \not< p$.

5. Let $X = \{1, \ldots, p\}$ be the set of the first p integers. The "positional", or "local" representation:

[2] If x is a real number, the notation $\lfloor x$ is used to refer to the smallest integer larger than x.

$$R_p: \begin{array}{ccl} X & \rightarrow & D = \Pi_{i=1}^{n}\{0,1\} \\ x & \mapsto & R_p(x) = (\underbrace{0,0,0,1}_{x},0,\ldots,0) \end{array}$$

is not a combinatorial representation, since again $n = p \not< p$.

6. Let X be any range of real numbers. Since X is uncountably infinite, X cannot be represented with a combinatorial representation.

3.2 Combinatorial domains

It appears clearly, from the few examples reviewed above, that the notion of combinatorial domain is intrinsically related to the representation used to describe the domain. Domains that we think of as combinatorial, then, are only so because we chose to represent them with combinatorial representations.

3.2.1 Definitions

The following definition of a combinatorial domain, thus, depends crucially on the interdependence of the nature of the domain thought of as combinatorial and the representation actually used to describe it.

Definition 2 A domain X is **combinatorial**, if and only if the representation R

$$R: \begin{array}{ccl} X & \rightarrow & D \subset \Pi_{i=1}^{n} A_i \\ x & \mapsto & R(x) = (d_{1_x}, \ldots, d_{n_x}) \end{array}$$

that is used for describing it or processing it is combinatorial.
A domain X is **purely combinatorial** if it is combinatorial and $R(X) = \Pi_{i=1}^{n} A_i$.
A domain X is **semi-combinatorial** if it is combinatorial and $R(X) \overset{\subset}{\neq} \Pi_{i=1}^{n} A_i$.

It is important to note, at this point, that although there are many different combinatorial representations that can be used to represent a domain, these representations are not necessarily of equivalent analytic value, in the sense that some representations are more useful than others in bringing to light the regularity or structure of the domain considered, if their is one.[3] The following examples illustrate this last point.

3.2.2 Examples

a. Let X be the domain of integers whose semi-combinatorial decimal representation S_d is:

$$S_d = \{26, 34, 35, 6, 70, 7, 78, 44, 25\}$$

(S_d is written here with the "concatenation" convention; that is, the decimal representation "26" stands for $(2,6) \in \{0,1,2,3,4,5,6,7,8,9\} \times \{0,1,2,3,4,5,6,7,8,9\}$).

X was not chosen with "random" integers, but was actually chosen with elements having some regularity. It is quite hard, however, to catch at first glance the regularity of the elements of X through the decimal representation shown above. If, however, X is now represented with the combinatorial representation mapping integers to their representations in base 3, we obtain the following set S_{b3}:

[3] A large part of the scientific inquiry can be seen as consisting in finding the appropriate representations and formalisms using these representations to describe the regularities of observed phenomena.

$$S_{b3} = \{0222, 1021, 1022, 0020, 2121, 0021, 2220, 1122, 0221\}$$

whose elements can, for clarity, be vertically arranged:

S_{b3} = {
0222,
1021,
1022,
0020,
2121,
0021,
2220,
1122,
0221
}

The regularity in X, which was hidden in the decimal representation, now appears clear: All elements of S_{b3} have the regularity that a 2 in the second right-most position of their representations in base 3 appears.

b. Now let X be a domain of integers whose combinatorial representation in base 3 yields the set $\tilde{S}_{b3}$ is:

$$\tilde{S}_{b3} = \{0121, 0000, 0022, 0220, 2222, 1012, 1210, 2002\}$$

Again, it is quite hard to capture the structure of X through the above representation. If, however, X is now represented with the combinatorial decimal representation yielding $\tilde{S}_d$,

$$\tilde{S}_d = \{16, 0, 8, 24, 80, 32, 48, 56\}$$

it appears that integers of X are all multiples of 4.
The non combinatorial "positional" representation of X yielding S_p, shown below, also displays, although maybe not as obviously, the structure of X:

S_p = {
................1..
1..
........1..
........................1..
....1
................................1..
....1................................
....1........................
}

It can be seen, then, that although some domains can be represented with various combinatorial and non-combinatorial representations, some representations are "better" than others, that is, catch the regularities of the domain in a more obvious way, for a given processing task like regularity recognition, as examples **a** and **b** showed.

The "goodness" of the representation depends of course both on the nature of the processing task, and on the nature of the domain. It can be conjectured, however, that domains that are "naturally" combinatorial, that is, domains that we have learned to describe and/or process with a given combinatorial representation, like language, [4] will, for just about every possible processing task, be better processed if they are represented with that combinatorial representation. Other domains will sometimes benefit from a combinatorial representation, but not always, as the preceding examples showed.

3.3 Combinatorial complexity

A combinatorial domain, that is, a domain represented with a combinatorial structure displays, by construction, some irreducible structure: Each "atomic" element in the sets A_i's will always appear at position i in the representation of an elements of the domain, and cannot appear anywhere else (unless another set $A_j, j \neq i$ contains the same atomic element).

If a domain is semi-combinatorial, additional constraints appear since each element of the sets A_i does not necessarily have to appear in any of the members of the domain. Different semi-combinatorial domains can thus have different degree of structure, depending on the nature of the constraints imposed on their elements. The two domains {abc, wbc, qbc, bck} and {akd, eod, wqx, ert}, for instance, do not have as much structure: The sub-pattern "bc" is always present in the elements of the first set, while there is not common sub-pattern in the second set. The set {abc, wbc, qbc, bck} is thus more redundant, or is less complex, than the set {akd, eod, wqx, ert}.

The word "complexity" will be used in this thesis to refer to how much regularity or structure a given represented domain has. Intuitively, the domain consisting of all written representations of English 4 letter words has more "structure" than the domain consisting of the same number of randomly chosen strings. Intuitively, also, the set of all English 4 letter words is more "complex" than the set of all English 3-letter words. Similarly, the set {abc, wbc, qbc, bck} has more structure, or is more redundant, than the set {akd, eod, wqx, ert}. In this section a quantitative definition for the notions of structure and redundancy is studied.

3.3.1 Entropy

To better characterize the notions of "complexity" or "structure", it is helpful to study a combinatorial domain from an information theory perspective. Intuitively, two domains whose individual elements can be fully described by different amounts of information have different complexities, and the concepts of "structure" and "redundancy" are linked with how statistically irregular, or uncertain, combinations of elements of the domain are. This observation was central in the development of information theory by Shannon (1948), whose important ideas are reviewed below.

Shannon's measure of information quantity

The observation that a source of information producing written English can be seen as a statistical generator of symbols where sequences of symbols depend on constant probabilities led Shannon to view a source of information as a statistically stationary, or stochastic, process. Furthermore, since written English as an information source has the property that definite probabilities of transitions between a symbol and the next exist, it can be viewed as a high-order Markov process, an ergodic one as transition probabilities are constant.

Having represented a source of information as a Markov process, Shannon then introduced a measure quantifying the expected amount of information per symbol produced by such a source. If S_n is any sequence of symbols produced by the source, and $P_n(S_n)$ the probability of occurrence of S_n, he showed that the function

$$E_n = -\frac{1}{n}\sum_{S} P_n(S_n)\log_2 P_n(S_n)$$

the entropy per symbol of the source approximated by an nth order Markov process, can be used to measure the expected amount of information, or uncertainty, conveyed by an nth order Markov process source, and

[4] With the restricted definition of a combinatorial domain given earlier, language as the set of all possible grammatically well-formed sentences is not combinatorial nor semi-combinatorial, since that set is infinite. Any finite subset of language, however, is semi-combinatorial.

defined the true entropy as [5]

$$E = \lim_{n\to\infty} E_n$$

Shannon (1951) was then able to estimate the entropy E of English, by successively approximating English generation with Markov processes of increasing order n yielding entropies E_n.

It is informative to look at typical sequences produced by Markov processes of different orders approximating English, and see how regularity and "structure" increases as the order of the process increases and as the entropy per symbol decreases.

A 0th order Markov process approximation (all symbols of English, 26 letters and a space, are equally likely) yields an entropy per symbol $E_0 = -\sum_1^{27} \frac{1}{27}\log_2(\frac{1}{27}) = \log_2 27 \simeq 4.76$. An example of a sequence produced by such an information source is:
`xfoml rxkhrjffjuj zlpwcfwkcyj ffjeyvkcqsgxyd qpaamkbzaacibzlhjqd`

If symbols are produced independently but with frequencies of English text, then the calculation of $E_1 = -\sum_i p_i \log_2(p_i)$ (first order Markov process yields $E_1 \simeq 4.03$, and an example of a sequence is:
`ocro hli rgwr nmielwis eu ll nbnesebya th eei alhenhttpa oobttva nah brl`
We can see here that the example displays more "regularity", and has more redundancies, than the preceding example.

If all symbols are not produced independently, but their probabilities only depend on the preceding symbol (second order Markov process), then Shannon calculated that $E_2 \simeq 3.32$, an example of a sequence being:
`on ie antsoutinys are t inctore st be s deamy achin d ilonasive tucoowe at teasomare fuso tizin andy tobe seace ctisbe`

If now symbols are produced with their probabilities depending on the two preceding symbols, (third order Markov process), the value $E_3 \simeq 3.1$ is found, and an example of a sequence generated by such a process is:
`in no ist lat whey cratict froure birs grocid pondenome of demonstures of the reptagin is regoactiona of cre`
Regularity is striking, as some of the blank-separated substrings "could" be English words. [6]

Entropy and Combinatorial domains

It is important to note at this point that although information entropy as defined by Shannon measures remarkably well the amount of statistical regularity in a stochastic information source, it fails to do so if it is blindly applied to a finite domain, like a combinatorial one: If $D \subset \Pi_{i=1}^{n} \tilde{A}_i$ is a combinatorial domain, for instance, its entropy $E(D)$ can naturally be defined as:

$$E(D) = -\sum_{a_1,\ldots,a_n} P(a_1,\ldots,a_n)\log_2 P(a_1,\ldots,a_n) \tag{3.1}$$

$$= \sum_{i=1}^{|D|} -\frac{1}{|D|}\log_2\frac{1}{|D|}$$

[5] If an event occurs with probability p, a measure $E(p)$ of how much surprise, or how much information is gained when the event actually occurs should have the properties that $E(1) = 0$ (an event happening with probability 1 creates no surprise, or brings no information), $E(p)$ is a strictly decreasing function of p (the smaller the probability, the greater the surprise, or amount of information gained), E is a continuous function of p (when p infinitesimally changes, so should $E(p)$), and $E(pq) = E(p)+E(q)$, (the surprise, or amount of information, caused by or gained from the joint occurrence of two events should be the sum of the surprises caused by, or information gained from, the independent occurrence of each event). Shannon showed that the only functions satisfying the 4 conditions listed above are functions of the form $E(p) = -C\ln(p)$, where C is a positive constant and determines the measuring unit of E. C can be chosen so that $E(p) = -\log_2(p)$, where $\log_2$ is the logarithm function of base 2. In this case, E is measured in bits. It follows that $E(p) = -\sum_i p_i \log_2(p_i)$, the entropy of X, measures the expected surprise, or uncertainty, of a discrete variable X that can take values x_i with probabilities p_i (See also Ross, 1988, for instance).

[6] Shannon found lower and upper bounds of 0.6 and 1.3, respectively, for E, when written English text was approximated by a Markov process of order 100. Later studies using different statistical techniques confirmed these results, with an estimate of $E \simeq 1.3$ bits per symbol by (Cover and King, 1978) and a estimate of $E = 1$ by (Grassberger, 1989).

$$= \log_2 |D|$$

where $a_1, \ldots, a_n$ are discrete random variables which can take as values any of the elements of $A_1 \subset \tilde{A}_1, \ldots, A_n \subset \tilde{A}_n$, respectively, and $P(a_1, \ldots, a_n)$ is the joint probability of occurrence of $a_1, \ldots, a_n$.

$E(D)$ thus depends solely on the number of elements of D, and although it clearly takes into account the fact that not all members of $\Pi_{i=1}^n \tilde{A}_i$ are in D, it is a rather crude measure of the complexity of D: It does not take into account the "internal structure" of D and is the same for any domain with the same number of elements. The set {abc, wbc, qbc, bck}, for instance, has the same entropy $E = \log_2 4 = 2$ as the set {akd, eod, wqx, ert}. Even worse, both of these sets have the same entropy as the set { skdjfhsk, mkocdsdc, sdkfsdf, qazpldjh }.

The problem here is that a well-defined measure that can be applied to stochastic processes "sequentially" generating an infinite number of symbols with fixed probabilities was roughly generalized to a domain that can be described and analyzed in terms of symbol frequencies but that cannot be modeled as a probabilistic Markov process, nor a fortiori as a stochastic process. To quote Kolmogorov (1965):

> The probabilistic approach [to a quantitative definitions of "information"] is natural in the theory of information transmission over communication channels carrying "bulk" information consisting of a large number of unrelated or weakly related messages obeying definite probabilistic laws. In this type of problem there is a harmless and (in applied work) deep-rooted tendency to mix up probabilities and frequencies within a sufficiently long time sequence (which is rigorously justified if it is assumed that "mixing" is sufficiently rapid). In practice, for example, it can be assumed that the problem of finding the "entropy" of a flow of congratulatory telegrams and the channel "capacity" required for timely and undistorted transmission is validly represented by a probabilistic treatment even with the usual substitution of empirical frequencies for probabilities. [...] But what meaning is there, for example, in asking how much information is contained in "War and Peace"? Is it reasonable to include this novel in the set of "possible novels", or even to postulate some probability distribution for this set?

3.3.2 Combinatorial complexity

There are at least two approaches that can be taken to the problem of measuring quantitatively the regularity or structure of a finite object. The first one is algorithmic in nature, and consists in viewing the description of an object as the output of a descriptive algorithm. Kolmogorov (1965), for instance, defined a measure of algorithmic complexity for a finite object as the minimum number of bits (the shortest program) containing all information about it sufficient for its decoding. (A clear survey of the main concepts of algorithmic complexity can be found in (Zvonkin and Levin, 1970).) A related measure of complexity was also introduced by Solomonoff (1964) (1978), Chaitin (1965) [7] and Lempel and Ziv (1974). [8]. Rissanen (1986) (1989) later refined these measures by introducing the concept of minimum length description.

Such measures of algorithmic complexity, or variants thereof, could possibly be used to generalize the measures which will be introduced in this chapter for the case of recursive laws of combination, as such laws and the expressions they allow can more easily be expressed in algorithmic notations. Given a grammar G and an alphabet Σ, for instance, a simple algorithmic measure of the complexity of the set of gramatically well-formed sentences with respect to G over Σ could be the length of the smallest deterministic automaton accepting these sentences, for instance (Ehrenfeucht, 1991).

Another approach, which has been used in psychology (Attneave, 1959), (Garner, 1962), (Staniland, 1966), is to further decompose the term

$$E(D) = - \sum_{a_1,\ldots,a_n} P(a_1, \ldots, a_n) \log_2 P(a_1, \ldots, a_n)$$

[7] The Shannon measure of complexity was related to measures based on Kolmogorov and Chaitin measures by (Leung-Yan Cheong and Cover, 1978) and later by Abu-Mostafa (1986), who showed that they were equivalent for a wide range of problems.

[8] It is interesting to note that the study of Lempel and Ziv concluded in the now famous LZ compression algorithm (Ziv and Lempel, 1978), a variant of which is used for the program "compress" in UNIX systems.

This second approach is concretely much simpler than the first for the case of combinatorial domains formed by concatenative laws of combination, will thus be the only approach taken here. [9]

Cartesian product of two sets
The case of a combinatorial domain $D \subset \tilde{A_1} \times \tilde{A_2}$, where

$$E(D) = -\sum_{a_1,a_2} P(a_1,a_2) \log_2 P(a_1,a_2) = \log_2(|D|)$$

is first studied.

Denoting by $P_{a_1}(a_2)$ the conditional probability of a_2 given a_1, and since $P(a_1,a_2) = P(a_1)P_{a_1}(a_2)$, E_D can be further decomposed:

$$E(D) = -\sum_{a_1,a_2} P(a_1,a_2) \log_2 P(a_1,a_2)$$

$$= -\sum_{a_1,a_2} P(a_1)P_{a_1}(a_2) \log_2 P(a_1)P_{a_1}(a_2)$$

$$= -\sum_{a_1} P(a_1) \log_2 P(a_1) - \sum_{a_1} P(a_1) \sum_{a_2} P_{a_1}(a_2) \log_2 P_{a_1}(a_2)$$

$$= E(A_1) + E_{A_1}(A_2) \tag{3.2}$$

where $E(A_1) = \sum_{a_1} P(a_1) \log_2 P(a_1)$ is the entropy of A_1 and $E_{A_1}(A_2)$, which has been called a conditional entropy by Shannon, measures the average entropy of A_2 for each value of elements of A_1. $E_{A_1}(A_2)$ can be interpreted as the "information" about A_2 from A_1.
It is important to note that

$$E(D) \leq E(A_1) + E(A_2) \tag{3.3}$$

The entropy of D is always smaller than the sum of the individual entropies of A_1 and A_2, with equality if and only if a_1 and a_2 are independent. This intuitive result can be proved by the following argument (Young, 1971):
If a_1 and a_2 are independent, then

$$E(D) = -\sum_{a_1,a_2} P(a_1,a_2) \log_2 P(a_1,a_2)$$

[9] Another measure that has not be investigated here but that possibly could have been used (Ehrenfeucht, 1991) is the Kullback measure of information K (Kullback, 1959) (Hobson and Cheng, 1973), defined as $\sum_i p_i \log_2 \frac{p_i}{q_i}$. K represents the information gained from the result of a random experiment when the prior probability q is changed to p.

$$= -\sum_{a_1}\sum_{a_2} P(a_1)P(a_2)\log_2 P(a_1)P(a_2)$$

$$= -\sum_{a_1}\sum_{a_2} P(a_1)P(a_2)\log_2 P(a_1) - \sum_{a_1}\sum_{a_2} P(a_1)P(a_2)\log_2 P(a_2)$$

$$= -\sum_{a_1} P(a_1)\log_2 P(a_1)\{\sum_{a_2} P(a_2)\} - \{\sum_{a_1} P(a_1)\}\sum_{a_2} P(a_2)\log_2 P(a_2)$$

$$= -\sum_{a_1} P(a_1)\log_2 P(a_1) - \sum_{a_2} P(a_2)\log_2 P(a_2)$$

$$= E(A_1) + E(A_2)$$

If a_1 and a_2 are not independent, then any prior information about the relation of an element of A_2 and an element of A_1 will reduce the surprise, or the information gained, when an element of A_2 appears with an element of A_1. In this case the information gained by the occurrence of an element of A_2 when an element of A_1 has already occurred is strictly less that the information which would have been gained if there was no relation between the two elements. It follows, thus, that

$$E_{A_1}(A_2) \leq E(A_2) \tag{3.4}$$

with equality if and only if a_1 and a_2 are independent. (An analytical proof of the inequality of equation 3.4, following (Ross, 1988), is given in appendix A)
The inequality of equation 3.3 follows from equations 3.2 and 3.4.

Since equation 3.3 holds, a symmetrical measure of the "inter-entropy" of the relation between A_1 and A_2, $R(A_1, A_2)$, can be introduced:

$$R(A_1, A_2)$$

$$= E(A_2) - E_{A_1}(A_2)$$

$$= E(A_1) - E_{A_2}(A_1)$$

$$= R(A_2, A_1)$$

$E(D)$ can then be rewritten as:

$$E(D) = E(A_1) + E(A_2) - R(A_1, A_2)$$

$E(D)$ is thus the sum of two terms, $E(A_1) + E(A_2)$ measuring, given the probability distribution induced by a_1 and a_2, the maximum entropy that could be obtained, and R measuring the "residual entropy" of $A_1 \times A_2$, due to the statistical dependency of a_1 and a_2. It should be noted that $E(A_1) + E(A_2)$ is not necessarily the maximum entropy that can be obtained given the symbols in $A_1 \times A_2$: Such a maximum entropy would be obtained if all symbols in each set A_1 and A_2 were equi-probable. In this case the equality $E(D) = \log |A_1||A_2|$ would hold.

General case

The symmetrical measure $R(A_1, A_2)$ of the "closeness" of the relation between A_1 and A_2 can be generalized for the cases of domains $D \subset \Pi_{i=1}^{n} A_i$ with $n > 2$. Since

$$P(a_1, \ldots, a_n) = P(a_1) P_{a_1}(a_2) \times \ldots \times P_{a_1, a_2, \ldots, a_{n-1}}(a_n)$$

where $P_{a_1 a_2 \ldots a_{i-1}}(a_i)$ is the conditional probability of occurrence of a_i given $a_1, \ldots, a_{i-1}$, $E(D)$ can be rewritten as

$$E(D) = E(A_1) + E_{A_1}(A_2) + \ldots + E_{A_1, \ldots, A_{n-1}}(A_n) \tag{3.5}$$

$E_{A_1, \ldots, A_{i-1}}(A_i)$ is the conditional entropy of A_i given $\Pi_{j=1}^{i-1} A_j$, and measures the average amount of uncertainty, or entropy, that remains in A_i when information on $\Pi_{j=1}^{i-1} A_j$ is known. The term $R(A_1 \ldots A_{i-1}, A_i)$ can be defined as

$$R(A_1 \ldots A_{i-1}, A_i) = E(A_i) - E_{A_1, \ldots, A_{i-1}}(A_i)$$

(with $R(A_1 A_0, A_1) = 0$) and the following equality holds:

$$E(D) = \sum_{i=1}^{i=n} E(A_i) - \sum_{i=1}^{i=n} R(A_1 \ldots A_{i-1}, A_i) \tag{3.6}$$

Since the maximum joint entropy of D, measuring the entropy of the random variables a_1 and a_2 when they are independent, is: [10]

[10] Since

$-\sum_{a_1, \ldots a_n} \Pi_{i=1}^{n} P(a_i) \log_2 (\Pi_{i=1}^{n} P(a_i))$

$= -\sum_{a_1} \sum_{a_2} \cdots \sum_{a_n} \Pi_{i=1}^{n} P(a_i) \log_2 (\Pi_{i=1}^{n} P(a_i))$

$= -\sum_{a_1} \sum_{a_2} \cdots \sum_{a_n} \Pi_{i=1}^{n} P(a_i) \sum_{i=1}^{n} \log_2 P(a_i)$

$= -\sum_{a_1} \sum_{a_2} \cdots \sum_{a_n} \Pi_{i=1}^{n} P(a_i) \log_2 P(a_1) - \ldots - \sum_{a_1} \sum_{a_2} \cdots \sum_{a_n} \Pi_{i=1}^{n} P(a_i) \log_2 P(a_n)$

$= -\{\sum_{a_1} P(a_1) \log_2 P(a_1)\} \sum_{a_2} P(a_2) \ldots \sum_{a_n} P(a_n) - \ldots -$

$\sum_{a_1} P(a_1) \ldots \sum_{a_{n-1}} P(a_{n-1}) \{\sum_{a_n} P(a_n) \log_2 P(a_n)\}$

$= -\sum_{a_1} P(a_1) \log_2 P(a_1) - \sum_{a_2} P(a_2) \log_2 P(a_2) - \ldots - \sum_{a_n} P(a_n) \log_2 P(a_n)$

$= \sum_{i=1}^{i=n} E(A_i)$

$$E_{max}(D) = -\sum_{a_1,\ldots a_n} \Pi_{i=1}^{n} P(a_i) \log_2(\Pi_{i=1}^{n} P(a_i)) \quad \text{or}$$

$$E_{max}(D) = \sum_{i=1}^{i=n} E(A_i) \tag{3.7}$$

$E(D)$ can be rewritten as:

$$E(D) = E_{max}(D) - \sum_{i=1}^{i=n} R(A_1 \ldots A_{i-1}, A_i) \tag{3.8}$$

or

$$E(D) = E_{max}(D) - R(D) \tag{3.9}$$

$E(D) = \log_2 |D|$ is thus the sum of two terms, $E_{max}(D)$, which can be called the combinatorial complexity $C(D)$ of D, measuring, given the probability distribution induced by $a_1, \ldots, a_n$, the maximum entropy that could be obtained, and $R(D)$ measuring the "residual entropy", or residual complexity of D, due to the statistical dependency of $a_1, \ldots, a_n$.

3.3.3 Examples

Table 3.1 lists values of entropy, combinatorial complexity and residual entropies, or complexities, as defined above for various sets including the sets A^n of all strings of the alphabet A = {a, ..., z} of length $n = 2, 3, 4, 5$ and 6, the sets W_n of English words of length n, and random subsets S_n of A^n having the same number of elements as W_n. It is observed as expected that the combinatorial complexity is always larger for a given set S_n than for the corresponding set W_n, the set of English words of length n being more "structured" than the corresponding set that has the same number of elements but that is composed with random strings. The residual complexities of the sets A^n is 0, since the probabilities of each symbol at a given position are independent.

Figure 3.1 graphically compares the combinatorial complexity E_{max} of the sets A^n, W_n and S_n, and shows again that the combinatorial complexity is always larger for a given set S_n than for the corresponding set W_n, the set of English words of length n

Figure 3.2 is a two-dimensional plot showing the same sets with their combinatorial complexity as ordinate and their residual complexities as abscissa. The point "5Lstrings, 3same letters" has also been added, showing the intermediate residual complexity between the set of all strings that have three letters in common, larger that the null residual complexity of the set of all stings of length 5 but smaller that the residual complexity of the set of all English 5 letter words.

3.4 Connectionist representations

The preceding study of combinatorial domains has introduced their definition, and a quantitative measure which can be used to discriminate between them according to their structure, in the context of **symbolic** representations: In this context, elements of a combinatorial domain are symbolically represented by the spatial concatenation of the symbolic representations of their atomic elements. Processing a combinatorial domain thus involves spatial symbol manipulation.

Table 3.1: Values of entropy, combinatorial complexity and residual complexity for various combinatorial and semi-combinatorial domains

Domain D	$E(D)$	$C(D)$	$R(D)$
{ aa, ab, ac }	1.58	1.58	0
{ aa, ab, ba }	1.58	1.84	0.25
{ aa, ba, bc }	1.58	1.84	0.25
{ aa, ab, ac, ad, ae }	2.32	2.32	0
{ ab, cd, ef }	1.58	3.17	1.58
45 randomly chosen different strings from an alphabet of length 8	5.49	5.57	0.08
45 English 2 letter words	5.49	7.57	2.08
45 different randomly chosen strings of length 2	5.49	8.89	3.40
All $26^2 = 676$ strings of length 2	9.40	9.40	0
753 English 3 letter words	9.54	12.34	2.78
753 different randomly chosen strings of length 3	9.54	14.02	4.47
All $26^3 = 17576$ strings of length 3	14.1	14.10	0
2177 English 4 letter words	11.09	16.03	4.94
2177 different randomly chosen strings of length 4	11.09	18.77	7.68
All $26^4 = 456,976$ strings of length 4	18.80	18.80	0
All strings of length 4 with 2 identical letters	16.58	18.80	2.23
3146 English 5 letter words	11.62	20.24	8.62
3146 different randomly chosen strings of length 5	11.62	23.47	11.85
All $26^5 = 11,881,376$ strings of length 5	23.50	23.50	0
All strings of length 5 with 3 identical letters	17.33	23.50	6.16
3852 English 6 letter words	11.91	24.09	12.18
3852 different randomly chosen strings of length 6	11.91	28.18	16.26
All $26^6 = 308,915,776$ strings of length 6	28.20	28.20	0

If a combinatorial domain is to be processed by a connectionist network, a connectionist representation mapping elements of the domain to vectors in a vector space needs to be found. Processing a domain now involves vectorial manipulations. In the rest of this chapter I will be concerned with the following question: Given a "symbolic" cognitive combinatorial domain D, what are the kinds of connectionist representations that are likely to preserve the structure of a combinatorial domain in such a way that connectionist processing of these representations is likely to display the same kind of properties, like generativity and systematicity, found in symbolic processing of the domain?

3.4.1 Trivial structure-preserving connectionist representations

A connectionist representation of a domain X is a mapping:

$$\mathcal{R}: \begin{array}{ccc} X & \rightarrow & V \subset R^m \\ x & \mapsto & \mathcal{R}(x) = v_x \end{array}$$

Let X be a combinatorial domain. X is thus naturally represented by a combinatorial representation R:

$$R: \begin{array}{ccc} X & \rightarrow & D \subset \Pi_{i=1}^{n} A_i \\ x & \mapsto & R(x) = (d_{1_x}, \ldots, d_{n_x}) \end{array}$$

Let $\mathcal{R}$ be a connectionist representation of D:

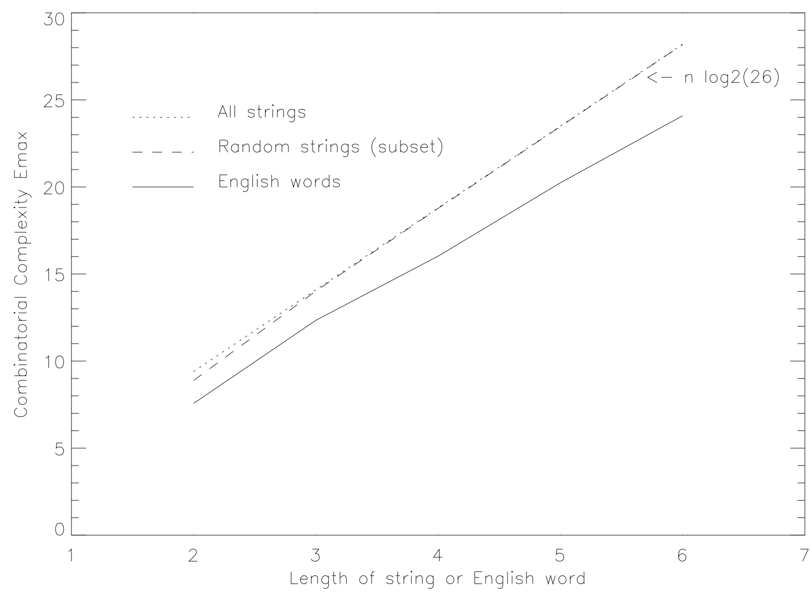


Figure 3.1: Graphical comparison of combinatorial complexity for some of the sets of English words and strings presented in table 3.1.

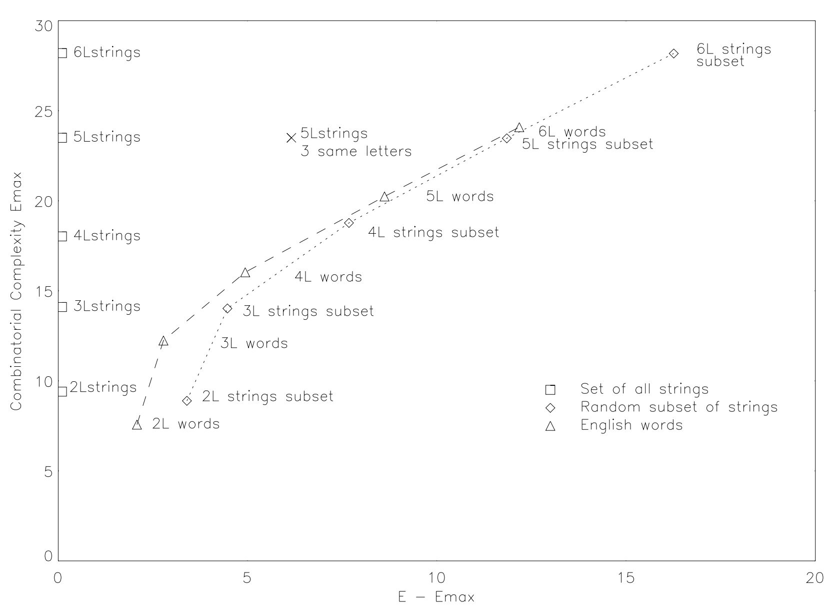


Figure 3.2: Graphical comparison of combinatorial complexity and redundancy for some of the sets of English words and strings presented in table 3.1.

$$\mathcal{R}: \begin{array}{lcl} D \subset \Pi_{i=1}^{n} A_i & \rightarrow & V \subset R^m \\ x & \mapsto & \mathcal{R}(x) = v_x \end{array}$$

In a symbolic representation of a combinatorial domain, elements of D are symbolically characterized by the spatial concatenation of their constituents in the sets A_i's. The combinatorial nature of $D \subset \Pi_{i=1}^{n} A_i$ is characterized by the facts that:

- For any element of D which has two or more identical constituents, any permutation of these same constituents yields that same element.
- For any two different members d_1 and d_2 of D that have i identical constituents in different positions, with $1 < i < n$, a permutation exists such that permuting these i constituents of d_1 yields d_2.

It is natural to conjecture that good connectionist representations for a combinatorial domain, that is, connectionist representation which would "code" the combinatorial systematicity of the objects of the domain and thus potentially lead to systematic processing, should exhibit these same properties.

In the case where the domain D is a Cartesian product of sets that have the same elements, it is possible to characterize such a constraint mathematically, by asking that for each permutation of atomic elements leaving a set of members of D unchanged, there exists a corresponding permutation of the coordinates in the space of connectionist representations of these members which does exactly the same, that is, that leaves unchanged the set of corresponding connectionist representations.

Formally:

Let S_D be the set of all permutations of n elements, and let S_V be the set of all permutations of m elements. For any $\sigma \in S_D$, let

$$I_D(\sigma) = \{d \in D, \sigma(d) \in D\}$$

$I_D(\sigma)$ is thus the set of elements of D which is invariant under σ.

Similarly, for any $\tau \in S_V$, let

$$I_V(\tau) = \{v \in V, \tau(v) \in V\}$$

Definition 3 $\mathcal{R}$ is a trivially structure preserving connectionist representation if and only if

$$\forall \sigma \in S_D,\ \exists \tau \in S_V,\ I_D(\sigma) = I_V(\tau)$$

3.4.2 Examples

Let D = {arc, bee, car, oak} $\subset$ {a,b,c,o} $\times$ {a,e,r} $\times$ {c,e,k,r}.

Then, $S_D = \{\mathcal{I}, (1,2), (1,3), (2,3), (1,2,3), (1,3,2)\}$, where $\mathcal{I}$ is the identity permutation[11], and

$$\begin{array}{lcl} I_D(\mathcal{I}) & = & D \\ I_D((1,2)) & = & \emptyset \\ I_D((2,3)) & = & \{\text{bee}\} \\ I_D((1,3)) & = & \emptyset \\ I_D((1,2,3)) & = & \{\text{arc, car}\} \\ I_D((1,3,2)) & = & \emptyset \end{array}$$

1. Let $\mathcal{R}_1$ be the connectionist semi-distributed tensor representation of D with fillers
a = (0,0,0,1), b = (1,0,0,1), c = (1,1,1,0), e = (1,0,1,0) k = (0,0,1,1),
o = (1,1,0,0) and r = (0,1,0,1), and positional roles
r_1 = (1,0,0), r_2 = (0,1,0), r_3 = (0,0,1).

[11] The other permutations are written using the cycle notation

$$\mathcal{R}_1 : \begin{array}{lcl} D & \rightarrow & V \subset R^{12} \\ d = (d_1, d_2, d_3) & \mapsto & \mathcal{R}_1(d) = d_1 \otimes r_1 + d_2 \otimes r_2 + d_3 \otimes r_3 \end{array}$$

The connectionist representations of the elements of D are:

$$\begin{array}{lcl} \mathcal{R}_1((a,r,c)) & = & (0,0,0,1,0,1,0,1,1,1,1,0) \\ \mathcal{R}_1((b,e,e)) & = & (1,0,0,1,1,0,1,0,1,0,1,0) \\ \mathcal{R}_1((c,a,r)) & = & (1,1,1,0,0,0,0,1,0,1,0,1) \\ \mathcal{R}_1((o,a,k)) & = & (1,1,0,0,0,0,0,1,0,1,0,1) \end{array}$$

$\mathcal{R}_1$ is trivially structure preserving.

Proof: For any $\sigma \in S_D = S_3$, let $\tau_\sigma \in S_V = S_{12}$ be the permutation

$$\tau_\sigma : \begin{array}{lcl} \{1,\ldots,12\} & \rightarrow & \{1,\ldots,12\} \\ i & \mapsto & \tau_\sigma(i) = 4\sigma((i-1)/4) + ((i-1)\mathbf{rem}4) + 1 \end{array}$$

where the sign " /" refers to integer division, and "rem" denotes the remainder operation.

τ_σ thus permutes consecutive blocks of 4 elements in the same relative positions as σ does. It follows that:

$$\forall \sigma \in S_D,\ I_D(\sigma) = I_V(\tau_\sigma)$$

2. Generalizing from the example above, it is easy to see that for any combinatorial X, any semi-distributed tensor representation using unit vectors of R^n as roles and different fillers for different elements of A_i is trivially structure preserving.

3. Let $\mathcal{R}_3$ be the following hashing representation of D:

$$\begin{array}{lcl} \mathcal{R}_3(\text{arc}) & = & (0,0,1) \\ \mathcal{R}_3(\text{bee}) & = & (0,1,0) \\ \mathcal{R}_3(\text{car}) & = & (0,1,1) \\ \mathcal{R}_3(\text{oak}) & = & (1,0,0) \end{array}$$

$\mathcal{R}_3$ is not trivially structure preserving. For instance, there does not exist any permutation of three elements $\tau \in S_V$ such that $I_V(\tau) = I_D((1,2,3)) = \{\text{arc, car}\}$, since the only permutations preserving $\mathcal{R}_3(\text{arc})$ are $\mathcal{I}$ and (1,2), but $I_V(\mathcal{I}) = \{\text{arc, bee, car, oak}\}$ and $I_V((1,2)) = \{\text{arc}\}$.

3.4.3 Structure-preserving connectionist representations

Definition **3** defined trivial structure-preserving connectionist representations: what was asked of these was that the spatial combinatorial properties found in symbolic representations of members of a combinatorial domain could be found on the coordinates of their representations.

It can be conjectured, however, that any connectionist representation which systematically transforms a trivially structure-preserving connectionist representations of a combinatorial domain will also "capture" the combinatorics of a combinatorial domain and thus exhibit the same kind of properties found in symbolic processing of combinatorial domains, or in other words be "structure preserving", or systematic.

The term "systematically" used above is unfortunately problematic here. We propose the following definition of a structure preserving connectionist representation, based on the existence of an invertible mapping between the structure preserving connectionist representation and any trivial-structure preserving connectionist representation.

A definition of an approximately structure preserving connectionist representation is also introduced.

Definition 4 $\tilde{\mathcal{R}}$ is a structure preserving connectionist representation of a combinatorial domain D if and only if there exists an invertible mapping $\mathcal{M}$ and a trivially structure preserving connectionist representation $\mathcal{R}$

$$\mathcal{R}: \begin{array}{lcl} D \subset \Pi_{i=1}^{n} A_i & \to & V \subset R^m \\ x & \mapsto & \mathcal{R}(x) = v_x \end{array}$$

such that

$$\forall x \in D, \tilde{\mathcal{R}}(x) = \mathcal{M}(\mathcal{R}(x))$$

Definition 5 Let $\|\ \|$ be the Euclidean norm on the representational space.
$\tilde{\mathcal{R}}$ is an approximate structure preserving connectionist representation of a combinatorial domain D, with precision p, if and only if there exists two mappings M_1 and M_2 and a trivially structure preserving connectionist representation $\mathcal{R}$

$$\mathcal{R}: \begin{array}{lcl} D \subset \Pi_{i=1}^{n} A_i & \to & V \subset R^m \\ x & \mapsto & \mathcal{R}(x) = v_x \end{array}$$

such that

$$\forall x \in D, \|\tilde{\mathcal{R}}(x) - M_1(\mathcal{R}(x))\| \leq p \ \text{ and } \ \|\mathcal{R}(x) - M_2(\tilde{\mathcal{R}}(x))\| \leq p$$

3.4.4 Examples

4. In the example above, any fully-distributed tensor representation $\tilde{\mathcal{R}_1}$ of $D = \{\text{arc, bee, car, oak}\}$ using the same fillers and linearly independent roles is structure preserving.

Proof: Let $\tilde{r_i}$, i = 1,2,3 be the linearly independent role vectors chosen, and let f_{x_i} be the vector representing the ith element of D in the decomposition $x = (d_{1_x}, \ldots, d_{n_x})$
There exists a matrix $M = \{\tilde{r_i}\}$, such that $\tilde{r_i} = Mr_i$, $i = 1,2,3$, (where the role vectors r_i are defined in section 3.4.2), since the vectors $\tilde{r_i}$ are linearly independent. It follows that:

$$\forall x \in D, \tilde{\mathcal{R}_1}(x) = \sum_{i=1}^{i=3} f_{x_i} \otimes \tilde{r_i} = \sum_{i=1}^{i=3} f_{x_i} \otimes Mr_i$$

The transformation T

$$\tilde{\mathcal{R}_1}(x) = \sum_{i=1}^{i=3} f_{x_i} \otimes r_i \quad \mapsto \quad T(\mathcal{R}_1(x)) = \sum_{i=1}^{i=3} f_{x_i} \otimes Mr_i$$

is linear in $\mathcal{R}_1(x)$. A matrix $\hat{M}$ thus exists, (it can be shown to be invertible, due to the fact that the fillers are lineally independent) such that

$$\forall x \in D, \tilde{\mathcal{R}_1}(x) = \hat{M}(\sum_{i=1}^{i=3} f_{x_i} \otimes r_i) = \hat{M}\mathcal{R}_1(x)$$

$\tilde{\mathcal{R}_1}$ is thus structure preserving, as $\mathcal{R}_1$ is trivially structure preserving as shown in section 3.4.2.

5. Let $\tilde{\mathcal{R}_2}$ be a connectionist RAAM representation of a combinatorial domain D. Then if a trivially structure preserving representation $\mathcal{R}$ is used to represent elements x of D on the input and output layers of the back-propagation network auto-associating $\mathcal{R}(x)$, $x \in D$, with a training criterion p, $\tilde{\mathcal{R}_2}$ is approximately structure-preserving, with precision p.

Proof: Let f be the activation function used in the auto-associative back-propagation network, M_1 and M_2 be the weight matrix of the first layer and second layer of the network, respectively, and, if $y \in R^q = (y_1, \dots y_q)$, let $F(Y) = (f(y_1), \dots, f(y_q))$. Then $\tilde{\mathcal{R}_2}(x) = F \circ M_1(\mathcal{R}(x))$, and $||\mathcal{R}(x) - f \circ M_2 \dashv \tilde{R}_2(x)|| \leq p$.

5. It is conjectured that the hashing representation $\mathcal{R}_3$ in example **3** is not structure preserving.

3.5 Conclusion

It is our intuition that the results reported in this thesis are crucially dependent upon the nature of the domain being processed and upon the kind of connectionist representations used, and can potentially be generalized to more complex tasks if and only if these involve elements of domains that present a combinatorial structure.

It is for this reason that such domains were studied in this chapter, as there is no doubt that a non-structure preserving connectionist representational scheme, such as a scheme based on hashing, could not possibly induce systematic or generative processing. Likewise, processing of a non-combinatorial domain could not possibly exhibit such properties. While the formalizations described are only a first step towards a full understanding of such domains, we believe that they do allow for a better understanding of the issues at play, which include:

- Availability of equally expressive representations for a given domain, yet non-equivalent if one is to perform inferences on them.
- Combinatorial complexity, central in its relation to the property of generativity, as will be seen in the next chapter.
- Structure-preserving representations, central in the hypotheses made in this thesis.

Chapter 4

Experiments with semi-distributed representations

4.1 Performance measures

As discussed in chapter 2, our measure of how well a particular network has induced the structure of the domain, when trained with examples of that domain, will consist in estimating how close the actual function f performed by the trained network is to an ideal binary functions f_B, as defined in section 2.4, which would correctly reproduce all members of the domain and incorrectly reproduce all non-members of the domain.

We could, to perform this estimation, compute an average generalization ability by computing the Hamming distance between f and a binary function $\tilde{f}_B$ which auto-associates perfectly every input bits of representations of all members of the domain, and incorrectly auto-associates (with maximum error) each input bits of representations of all non-members of the domain. This would involve testing the network on all possible binary input patterns, and counting how many output bits, for each of these binary patterns, are incorrect. Since such testing would involve, for n-bit patterns, $n2^n$ comparisons, this is clearly unfeasible: Even a middle sized network with 30 input units would require on the order of 3×10^{10} processing passes through the network.

Another way to perform that estimation, and one more closely related to classic measures of human performance used in cognitive science, would be to perform two measures, the first giving an indication on how well the network has learned members of the domain, and the second estimating how well the network has discriminated between members and non-members of the domain. This is what we chose. The first measure simply amounted to estimating the number of true and virtual generalizations produced by the network for members of the domain. The other measure was performed by generating a discrimination measure based on the number of true and virtual generalizations that a given network would produce for members of the domain, along with the number of incorrect true and virtual generalizations the network would produce for non-members of the domain. We used the statistical discriminability measure (Estes, 1982),

$$d = \log\left[\frac{P_X(1-P_{\bar{X}})}{(1-P_X)P_{\bar{X}}}\right]$$

where P_X is the probability that a member of X is accepted, and $P_{\bar{X}}$ the probability that a non-member is accepted. Values of 2-4 for such a discrimination measure correspond to reliable discriminability.

Since P_X and $P_{\bar{X}}$ are small, as we will see, d is close to $\log(P_X/P_{\bar{X}})$; d, then, is the number of orders of magnitude (based on e rather than 10) by which P_X exceeds $P_{\bar{X}}$. The measure d corresponds to an underlying logistic cumulative distribution, rather than a normal or Gaussian error function, which underlies the d' measure of discriminability. (The d' values for the discriminations performed by most of the networks we will report on were rather low, since, while $P_X/P_{\bar{X}}$ was high (as reflected in d), P_X itself was rather low.)

In attempting to characterize the number of true and virtual generalizations for a given n across networks with various random choices of weights and training sets, we have little a priori information about the distribution we are sampling. We have chosen to employ the median [1], because it provides a simple distribution-free analysis. Repeating experiments five times with different randomly chosen training sets and initial weights will ensure that our estimated median number of generalizations or virtual generalizations approximates the true number with a maximum error of around 6%, since, if the minimum of a given sample of five independent experiments is m, the maximum is M, and μ is the true median, then

$$\text{prob}(m > \mu) = (1/2)^5 = \text{prob}(M < \mu);$$

and

$$\text{prob}(m < \mu < M) = 1 - 2 \times (1/2^5) = .9375$$

That is, the range between the minimum (m) and maximum (M) obtained in five independent experiments gives an approximate 94% confidence interval for the median.

4.2 Outline of our experimental approach

We will, then, train a feed-forward auto-associative network on a randomly selected subset of the chosen domain, and test for correct association each of the remaining items of the domain, one by one. If an untrained test item is not generalized, we will train the network on it for a small number of trials and, if the test item is then learned to criterion, we then test each item of the original training set for correct association. If all items of the training set are still correctly associated after the fast training of the new test item, that item will not have interfered with the correct auto-association of items of the training set and we will call it a virtual generalization, as defined in the previous chapter. Although it was not correctly associated and thus was not generalized, it was "almost" generalized, since only a small number of trials, preserving performance of the items of the original memory intact, allowed it to be correctly associated.

Since we would like to show that connectionist learning and processing can be sensitive to the underlying structure of the representations of the objects being learned and processed, and can also allow for productivity, we will count (or infer statistically) how many of the test items are generalizations, that is, how many are correctly auto-associated, and how many are virtual generalizations. A large number of generalizations relative to the size of the training set, and explosive growth with the complexity of the domain, will show that processing is productive and systematic by exploiting the compositionality and systematicity of the representations. Fast interference-free learning of new items of the domain will show that learning exploits the compositionality and systematicity of the representations, while a large number of virtual generalizations obtained and their explosive growth with the complexity of the domain will, again, show productivity and systematicity.

4.3 Choice of experimental parameters

4.3.1 Domain sizes

To measure how performance grows with the complexity of the domain $X = \prod_{i=1}^{n} X_i$, we will perform experiments with n varying and plot the median number of generalizations and virtual generalizations obtained for members of the domain. If A is the size of each (identical) set X_i in X, then the number of elements in the domain , $||X||$, is A^n. A good experimental choice for A would be one that allows both a decent size for X for small values of n, and a small enough number, since the size of X grows exponentially with n, for larger n's to allow testing on a sizable fraction of the domain. We mean, by "decent" size for X for small values of n, a size that would allow training sets to be both small relative to X, (say, smaller by at least an order of magnitude), and large enough to reflect the statistical regularity of the domain.

[1] Clayton Lewis is gratefully acknowledged for suggesting this.

We found that the choice of $A = 26$, while allowing n to vary within the range [2,6], somewhat obeyed these constraints, when choosing a training set size of $p = 50$. That choice, furthermore, allowed comparisons with experiments involving English words.

With $n = 2$, networks were thus trained on random sample sets of 7 % of the 26^2 members of X. With $n = 6$, A^n was on the order of 10^8, and testing set samples of size $10^4 - 10^5$ allowed for an estimation of the median. Such testing set sizes were clearly an upper limit in terms of computational effort, as discussed in the next section.

4.3.2 Network sizes

Input and output layers.

With $A = 26$, the minimum number of bits to code an individual letter would be 5. We decided to use 8-bit fillers, thus giving each representation 3 extra bits to differentiate itself from others and allowing coding of domains with larger alphabets. 3 extra bits allow for at least $2^3 = 8$ times as many bit representations as legal string representations, which means that binary patterns representing members of the domain will constitute a fraction $(26\ /\ 2^8)^2 \approx 1\%$ of the total number of binary strings. Th sizes of the input and output layers of the networks were thus $8n$.

Hidden layer

The choice of the size of the hidden layer, relative to the size of the input and output layers, is a crucial one, since it directly affects generalization ability.[2] Intuitively, if a back-propagation network has too many hidden units, then the network will easily develop internal representations in the hidden layer, without relying on the statistical regularities of the training set, to allow for correct performance on the training set. Each training pattern will thus be learned "individually". If few hidden units are used, on the other hand, the network is forced to extract the statistical regularities of the training set to perform correctly on the training set. It is this extraction, then, which will allow for good generalization.

As was briefly mentioned in section 2.4.4, a number of techniques have been devised to reduce the number of free parameters (weights to hidden units or hidden units themselves) to improve generalization in a back-propagation network learning a given task. These include weight removal, where a weight "cost" term tending to penalize large weights is added in the error function on which gradient descent is performed, (Hanson and Pratt, 1989), (Chauvin, 1990) (Weigend *et al.*, 1990), unit removal, where a measure of unit relevance is assessed allowing irrelevant ones to be discarded, (Mozer and Smolensky, 1989), and network construction algorithms, (Mezard and Nadal, 1989) (Sirat and Nadal, 1990) (Marchand *et al.*, 1990) (Frean, 1990) (Sankar and Mammone, 1991), where networks are constructed with optimization in mind.

Since our purpose was not to study how networks with specialized architectures could best generalize for our chosen tasks, but was rather to study, with generality and simplicity in mind, generalization ability of "average" networks, we did not use any of these techniques in our main experiments (chapter 4 and chapter 5. Instead we simply empirically determined, by performing a number of preliminary experiments with varying hidden layers sizes, the greatest constant compression ratio, independent of n, which would allow learning of the training sets. A ratio of 8/5 turned out to allow, with some difficulty sometimes, learning of training sets [3]. The number of hidden units H was thus $5n$. When a particular network could not learn a training set, we simply started the experiment again with new random weights.

It should be noted, then, that the results obtained in our main experiments are in no way the best results possible, [4] in terms of generalization ability. They will, rather, indicate average performance. In chapter 6, the technique of weights elimination was used. The corresponding experiments, in that case, indicate optimum performance.

[2] An often heard rule of thumb is that the best generalization ability will be accomplished with the smallest network capable of learning the training set.

[3] It is interesting to note that 5 is the minimum number of bits to code all letters

[4] Yu and Simmons (1990) have argued, furthermore, that reduction of the total sum of error over all training patterns, as performed by the traditional gradient descent back-propagation algorithm, will not necessarily lead to a solution that is best in terms of binary output correctness, which is the measure we use. This could be seen, for example, in a network where a great majority of output units would be very close to their target value, while a few would be very far. The sum of squared errors, in that case, would still be small while the pattern would not have been learned according to our measure. Yu and Simmons (1990) have proposed a technique called "Descending Epsilon" to improve binary correctness ratio when back-propagation is used. That technique involves, at an extra computational cost, a "selective attention" on individual output units that are far from their target.

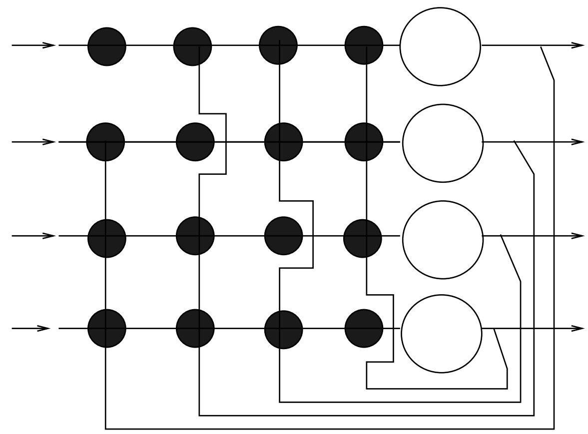


Figure 4.1: Architecture of the recurrent auto-associator with no self-connections, after McClelland & Rumelhart [1986]. Each of the 4 units (white) are connected to all others.

4.3.3 Computational requirements

If E is the number of floating point operations needed for the calculation of the sigmoid for a given point, then a pass through a network with N units in the input and output units, and H hidden units, requiring the computation of the squashed weighted sum of activities for each non-input unit, will require on the order of $C[H(2N+E-1) + N(2H+E-1)] = 4CNH + C(E-1)(N+H)$ floating point operations, where C is a constant of proportionality to be determined later.

With 8-bit fillers and n-bit orthogonal roles to code elements of a domain $X = \prod_{i=1}^{i=n} X_i$, then $N = 8n$. A compression ratio of 8/5 for the hidden layer yields $H = 5n$. Assuming an approximate $E = 45$ floating-point operations for the calculation of the sigmoid (1 division, 1 addition, and a power series expansion to approximately 8 terms, for the calculation of the exponential, which induces about 27 multiplications, 7 divisions by factorials found in hashing tables, and 8 additions), then a pass through the network will involve on the order of $C(160n^2 + 572n) = 160Cn(n + 3.6)$.

If p is the number of patterns in the training set and T is the number of patterns in the testing set, then testing for virtual generalizations will involve on the order of $160CTpn(n + 3.6)$ floating-point operations to perform the Tp passes needed to test the correct performance of patterns of the training set after a new pattern of the testing set is learned. (We ignore here the number of floating point operations required for the fast learning of each virtual generalizations, since it is small compared to the number of operations required to test correct performance of the original training set). Repeating an experiment 10 times with $T = 10{,}000$ testing patterns for a training set of size $p = 50$, (5 times to test members of the domain and 5 times to test random-bit patterns) will thus require on the order of $8Cn(n+3.6) \times 10^8$ floating point operations.

Small simulations yielded a value of about 15 for C. With $n = 6$, a machine performing at around 1 mflops such as a Sun 4 or Sparcstation 1 would then need on the order of a week to perform the approximate 7×10^{11} floating point operations needed for a median point. Use of a powerful 20-processor Encore Multimax machine, however, allowed us to reduce that time to an order of a day.

4.4 Early experiments

4.4.1 Structure of English words with a recurrent auto-associator

Some early experiments were performed not with a standard feed-forward back-propagation network, but with a non-linear recurrent auto-associator. (Kohonen, 1977) (Anderson *et al.*, 1977) (McClelland and Rumelhart, 1986), the architecture of which is shown in figure 4.1. Such highly recurrent networks (each unit is connected to all others) are typically used for modeling content-addressable memories: Training is performed using a variant of the delta rule, and during testing, or recall, a part of a pattern is presented to the network. Activity circulates in the connections for a number of cycles until equilibrium is reached, after which the entire pattern, hopefully, is reconstructed on the network's units.

Experiments involving 4-letter English words (section 4.4.3) were performed with such a network, but were abandoned for a number of reasons.

First, the network, lacking hidden units, was considerably less performant that an equivalent 3-layer standard back-propagation network. Training for large numbers of epochs (more than 5,000) was not enough to reduce overall error to the point where correct performance on the training set, with criteria similar to the one used with a standard back-propagation network, was achieved. Hidden units could have been used using recurrent versions of back-propagation (Almeida, 1987) (Pineda, 1987), but this would have raised computational costs which, as we have seen earlier, are high in our simulations.

Second, computation costs associated with testing untrained patterns were high, since there is more that one way to select and present "part" of a pattern. Relatedly, computational costs associated with testing the networks are high, because of the number of cycles needed to reach equilibrium.

Finally, there is no reason to believe that results obtained with our experiments cannot be generalized to recurrent networks, which have been shown (Almeida, 1987) to perform better at pattern completion.

4.4.2 The $XX' \cup YY'$ problem

Another experiment involved learning a domain whose structure reflected, in a trivialized way, grammatical determiner/noun agreement structure. Four sets X, Y, X' and Y' of random vectors, where $X \cap Y = \emptyset$ and $X' \cap Y' = \emptyset$, were generated, which could be interpreted as representing singular determiners, plural determiners, singular nouns and plural nouns, respectively.

$$
\begin{aligned}
X &= \{\texttt{"a", "this", "that", "one"}\} \\
Y &= \{\texttt{"some", "many", "all", "several"}\} \\
X' &= \{\texttt{"dog", "cat", "person", "student", "friend"}\} \\
Y' &= \{\texttt{"dogs", "cats", "persons", "students", "friends"}\}
\end{aligned}
$$

The network was trained, within the back-propagation auto-associative paradigm, on a subset of positive examples drawn from $XX' \cup YY'$, where each example was formed by concatenating the representations of members of X and X', or Y and Y'. The task was to correctly generalize or quickly learn with no interference all members of $XX' \cup YY'$, while discriminating them from non members, e.g. members of $XY' \cup YX'$.

The network was able to perfectly generalize and discriminate when the slightest indication (i.e an on or off bit) of either "singular" or "plural" was coded in members of X and X', and members of Y and Y', respectively. Such an indication [5] , of course, trivializes the problem, as the network only needs to learn to be a "bit" detector, that is, process that particular information to "decide" whether an association should be made or not.

A large number of experiments, within the back-propagation paradigm, were conducted without success, with representations lacking indications as described above. The network could easily learn the training set but, systematically generalizing on both $XX' \cup YY'$ and $XY' \cup YX'$, could not discriminate. It thus learned to recognize $(X \cup Y)(X' \cup Y')$, but not $XX' \cup YY'$. Several simulations involving various semi-local and distributed representations, hidden layer sizes, training and testing set sizes were performed, yielding the same conclusion. The networks, then, learned to be "too" systematic.

4.4.3 Regularity detection: English 4-letter words

We report here on an early experiment designed to test our basic assumption that a feed-forward auto-associator is capable of learning through back-propagation to recognize whether an unfamiliar sequence shares in the combinatorial regularities characterizing some domain.

[5] It should be noted that a corresponding indication exists in most, but not all, determiners and agreeing nouns in English, as an ending "s", for instance, in most commonly an indication of the plural form of a noun.

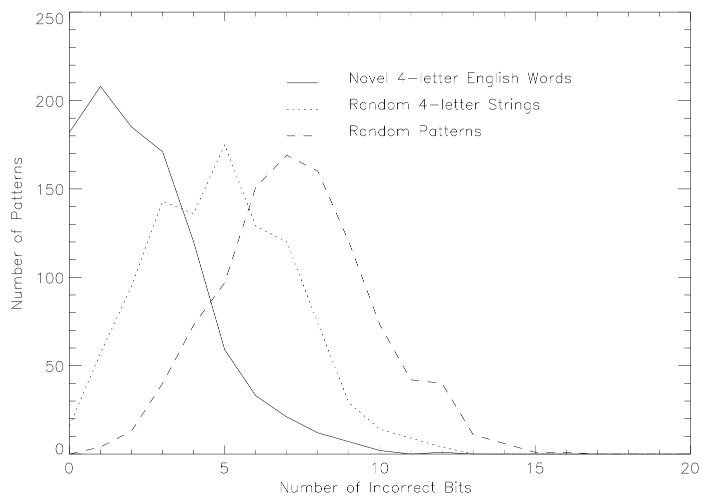


Figure 4.2: Generalizations; Network trained on 100 English 4-letter words.

In this experiment, we took the domain X to be a set of 1100 4-letter English words[6], and trained a network (here and henceforth, a back-propagation feed-forward $8n \times 5n \times 8n$ auto-associator, unless specified otherwise, where n is the length of the letter sequence) on 100 randomly selected such words. We then tested its generalization ability on the 1000 remaining untrained words, on 1000 randomly selected 4-letter strings, and 1000 random bit patterns. If the network can recognize the degree to which new patterns share in the regularities with the training set, it should generalize best with English words, then 4-letter strings, then random-bit patterns. This is confirmed experimentally in figure 4.2, where 87% of English words have less that 5 incorrect bits, versus 45% for random strings and 22% for random bit patterns.

Not only do new examples that share in the regularities of the training set produce fewer erroneous output bits, but they are also easier to learn, as shown in figures 4.3 and 4.4.

The discrimination measure d between English words and random-bit patterns was 3.7, indicating good discrimination. The value of d corresponding to discrimination between English words and 4-letter strings was much lower, at 1.12. Experiments reported in chapter 6 will show, however, that such a measure can be improved. In the following experiments we will summarize information on generalization and ease of learning of a new input by reporting just the number of generalizations (the number of novel patterns with zero incorrect bits) and the number of virtual generalizations. We observe (figure 4.4) that the number of generalizations and virtual generalizations is the biggest for the set of English words, then for the set of random 4-letter strings. For the random selection of 1000 random bit patterns, there were simply no generalizations.

4.5 Generalization

In this section we describe generalization results of our main experiments, addressing learning in Cartesian product domains $X = X_1 \times X_2 \times \ldots \times X_n$ for various values of n and sets X_i. For clarity, all results pertaining to virtual generalizations will be given in the next section.

For the cases $n = 2$ and $n = 3$, it was possible to test the entire set of untrained examples; the number of generalizations (and virtual generalizations in the next sections) for these two cases is thus exact. For the cases $n = 4, 5, 6$, complete testing was not feasible for computational time reasons (section 4.3.3). We therefore tested a sample of T randomly-generated examples (with replacement). The number of generalizations plotted (and later, the number of virtual generalizations) is thus an estimate, assuming unbiased samples. For each value of the varying parameter (in this case n), we repeated our experiments 5 times (as in all subsequent

[6] The words used were an arbitrarily chosen subset of the most common English 4-letter words.

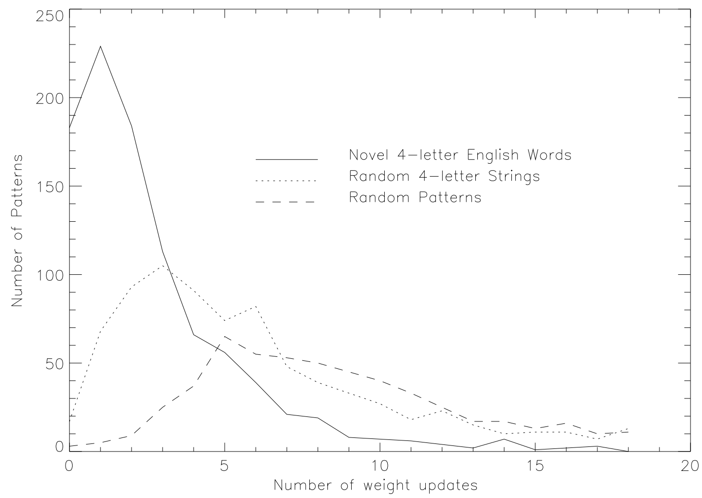


Figure 4.3: Number of weight updates to learn a new input, after training on 100 4-letter English words.

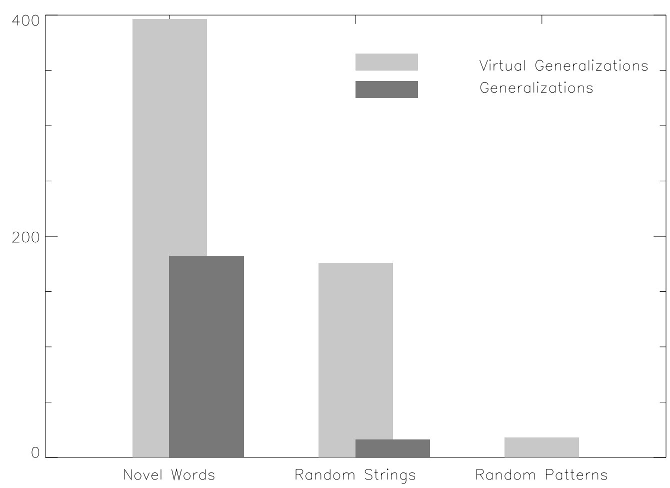


Figure 4.4: The number of generalizations and virtual generalizations for the network trained on 100 English 4-letter words.

experiments), each time starting with new random initial weights, a new randomly selected training set, and a new randomly selected testing set of T patterns in the cases n = 4, 5, 6. For n = 4 and 5, T was 10,000. For $n = 6$, T was 100,000 for generalizations and 10,000 for virtual generalizations. Table 4.1 summarizes this information. All figures display the number of generalizations (or virtual generalizations) obtained for each experiment, as well as the line connecting the median number of generalizations (or virtual generalizations) obtained. Discriminations measures are shown in the following subsection.

Table 4.1: Sizes of the testing sets used in experiments.

n	Domain X Size	Testing Set Size
2	$676 = 26^2$	676
3	$17576 = 26^3$	17576
4	$456{,}976 = 26^4$	10^4
5	$11{,}881{,}376 = 26^5$	10^4
6	$308{,}915{,}776 = 26^6$	10^5

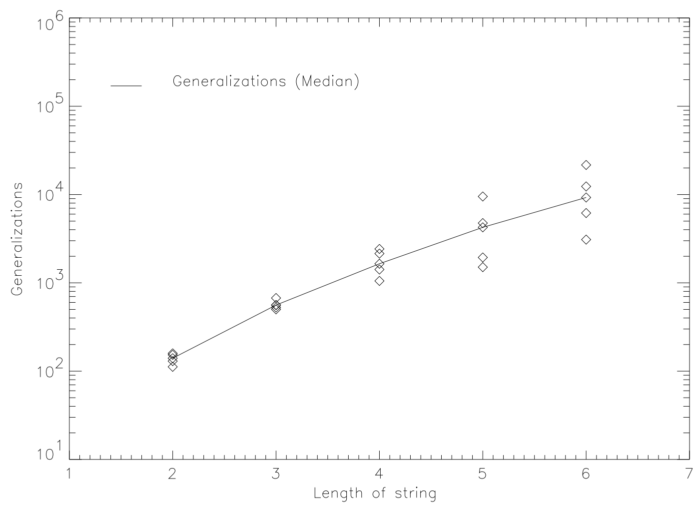


Figure 4.5: Exponential growth of generalizations for networks trained on sets of size 50, with $A = 26$, as n varies from 2 to 6.

4.5.1 Main results

The number of generalizations in networks trained on $p = 50$ examples in the case $A = |X_i| = 26$ and $n =$ 2,3, ..., 6 is shown in figure 4.5, in a semi-logarithmic plot in which a straight line corresponds to the graph of an exponential function. As n increases and the combinatorial complexity of the domain increases (as shown in figure 4.6 [7]), we observe that the number of generalizations grows approximatively exponentially.

Although, with the exception of domains with a small degree of combinatorial structure, or combinatorial complexity, ($n \leq 3$), the networks have been trained on a very small fraction of examples, the number of generalizations outnumber the number of training examples by several orders of magnitude. We estimate, for instance, over 4,000 generalizations in the case $n = 5$, although the training set size (50) corresponds to 4×10^{-4}% of the domain. For $n = 6$, we estimate 10,000 generalizations, with a training set size corresponding to 2×10^{-5}% of the domain.

Figure 4.7 shows the number of generalizations as the size of the sets X_i's– the alphabet size A–, varies, for $n = 4$ and training sets of size p. The overall trend here seems to suggest that the number of generalizations decreases with the size A of the alphabet, a behavior which could be explained by the following tentative

[7] These complexity measures, and the ones that follow, were performed on the "written" representations of the strings. What the network is exposed to, of course, are their bit representations. Corresponding measure on the bit representations, however, did not reveal qualitative differences in the shape of the curves.

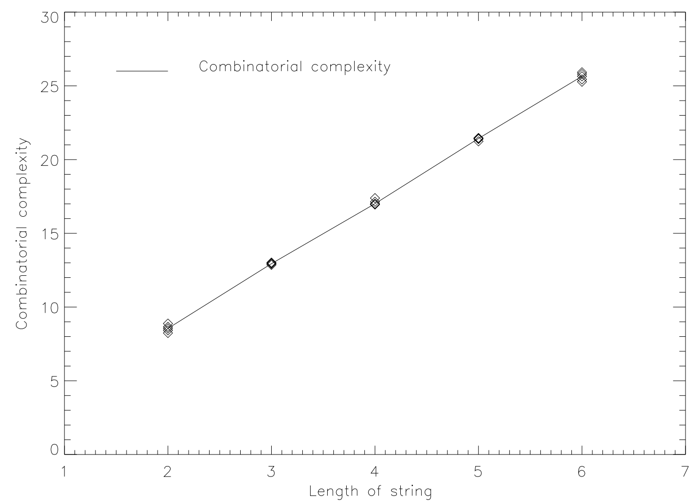


Figure 4.6: Combinatorial complexity of the domain X as n increases.

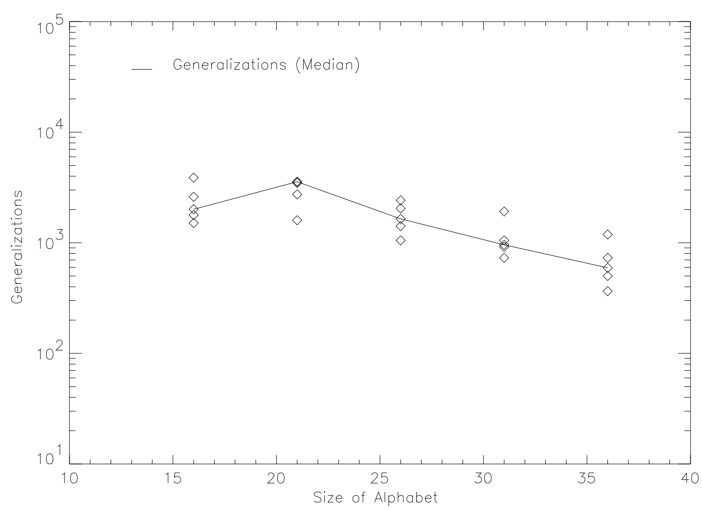


Figure 4.7: The number of generalizations for networks trained on sets of sizes 50, with $n = 4$, and $A = 16$, 21, 26, 31, 36.

account involving two opposite trends.

On one hand, since a letter, for a given position, is seen with a probability $50/A$ on average during training, its frequency during training is inversely proportional to A. Due to this factor, performance, then, could be hypothesized to decrease as A increases, for a fixed domain size. [8] The domain size A^4, on the other hand, grows polynomially as a function of A. We could thus hypothesize, for a fixed performance, a growth in generalizations as A increases.

What figure 4.7 seems to suggest, then, if these two hypotheses are correct, is that performance decreases as presentation frequencies of a given letter at a given position decrease, less rapidly than the opposite growth

[8] Such a performance decrease is in fact, although in a slightly different context, observed in figure 4.10 where the number of generalizations is shown as a function of the size of the training set.

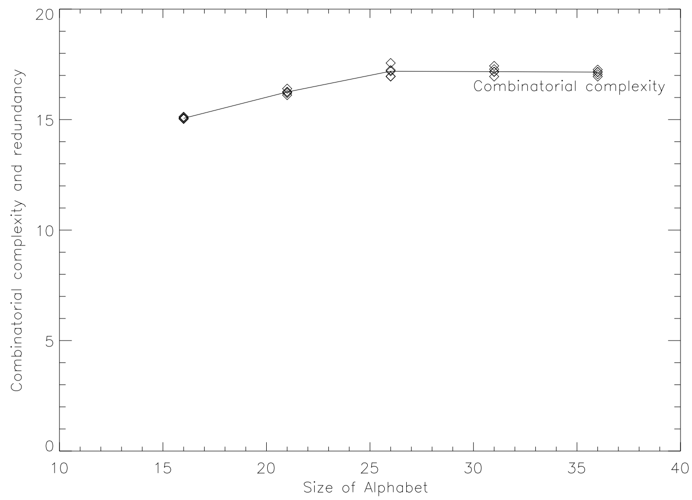


Figure 4.8: Combinatorial complexity of the domain X as A increases.

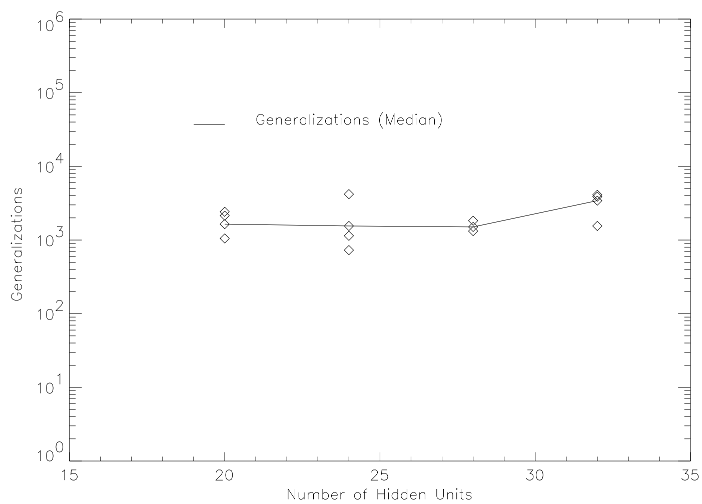


Figure 4.9: The number of generalizations for networks trained on sets of sizes 50, with $n = 4$, $A = 26$, as H varies

in generalizations due to polynomial growth of the domain with A at first, but more rapidly afterwards.

Performance thus follows closely the combinatorial complexity of the training set. This is supported by figure 4.8, which shows how the combinatorial complexity of the domain X evolves as A increases. We observe that the shape of the curve crudely approximates the shape of the curve of figure 4.7.

In figure 4.9, the relation between the number of generalizations obtained and the size H of the hidden layers used in the networks is shown, as $n = 4$ and $p = 50$ are constant. The plotted points correspond to compression ratios $N : H$ from input to hidden units of 8:5, 8:6, 8:7 and 1. The number of generalizations is surprisingly rather insensitive to these, although an expected increase, as the compression ratio approaches 1, occurs: When the number of hidden units is sufficiently close to the number of input units (and the equal number of output units), the networks can more easily compute an approximation of the identity function,

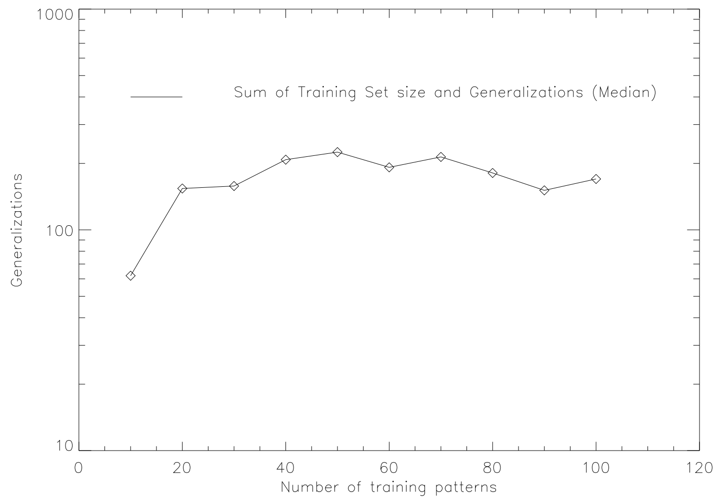


Figure 4.10: Sum of training set sizes and generalizations for networks trained on sets of sizes p, with $n = 2$, $A = 26$, as p varies

instead of relying on the combinatorial structure of the domain. The networks, clearly, never compute such an identity mapping, since performance is never optimal on the domain. No matter how many hidden units are present, then, they rely on the statistical regularities of the training examples. But adding more hidden units decreases this reliance, thus inducing more generalizations. This is also indicated by the lower left graph of figure 4.12, where discrimination is seen to decrease between $H = 20$ and $H = 25$, and although it rises again, it never reaches it first value estimated at around 4.

Combined with the present graph, figure 4.12 concerning discrimination can help us develop the following very tentative account: When the number of hidden units is small ($H = 20$), the network discriminates well, with an estimated value of $d = 4$, as the structure of the domain is relied on. As H increases, discrimination first decreases to around 2 as the combinatorial structure of the domain is less relied upon, but the networks do not generalize better since the added hidden units have had the effect of both increasing generalization power, (the networks implement the identity function closer) and decreasing generalization power (the networks rely less on the structure of the domain). As H approaches 32, the size of the input and output layers, generalization finally increases as the first effect overcomes the second. Discrimination increases again, to an estimated value of about 3 (smaller that the first value of 4), as generalizations of patterns of the domain grow slightly faster than generalizations of non-members of the domain. Such an analysis, we should point out, is highly tentative, given the small number of data points available.

Figure 4.10 shows the sum of training set sizes and generalizations for networks trained on p patterns as p varies, with $n = 2$ and $A = 26$ constant. While training patterns were not counted as generalizations earlier, we include them here since the number of generalizations is small with respect to p, and since this graph can be better interpreted as showing performance growth, or generalization ability, as a function of training set sizes. The trend here indicates exponential growth for small values of p, asymptoting as training set sizes get large. For small training set sizes, performance increases quickly as induction of the combinatorial structure of the domain occurs. Past a certain value (around $p = 30$), this induction is mostly done, and generalization ability does not increase as quickly, the networks having closely reached, with their limited capacity due to their small sizes, their optimal performance for the given task. These results are in accordance with general theoretical studies of generalization ability, as discussed in the previous chapter.

Figure 4.11 shows how combinatorial complexity evolves with the size of the training sets. The shape of the curve, again, approximates that of figure 4.10.

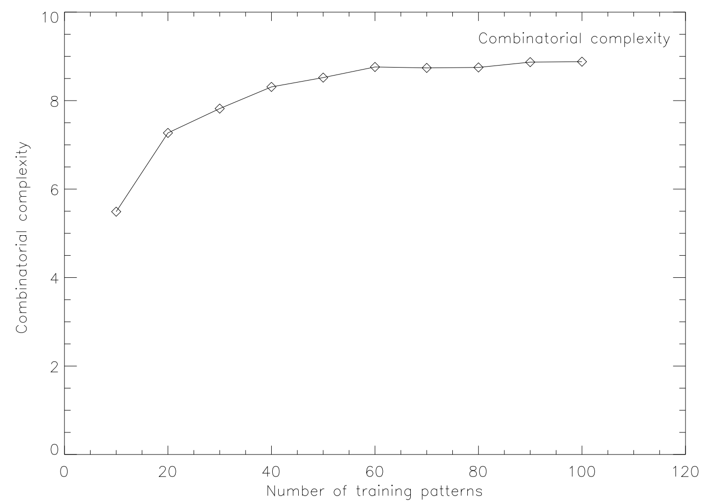


Figure 4.11: Combinatorial complexity of the domain X as p increases.

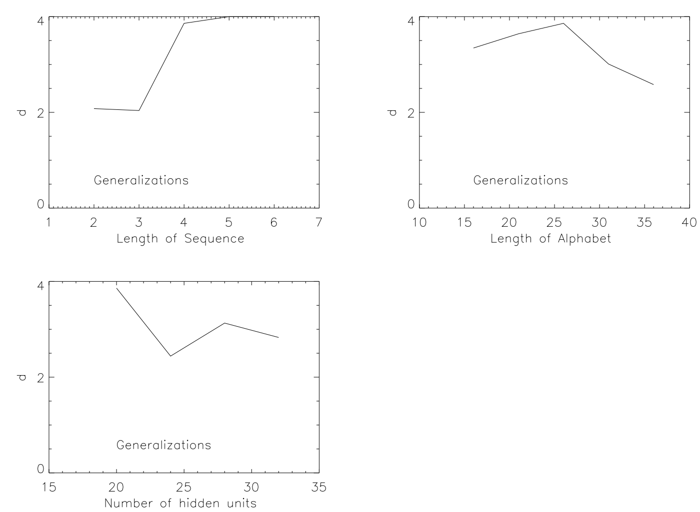


Figure 4.12: Discrimination measures for generalizations

4.5.2 Discrimination tests

As was mentioned in chapter 2, discrimination measures are crucial in our experiments, since no discrimination would indicate that the networks have failed to induce the combinatorial structure of the domains, by generalizing equally on both members and non-members of the domain.

Figure 4.12 show estimates of such measures, obtained by testing random binary patterns in the same conditions present for testing members of the domain. [9] The upper left graph, showing discrimination for

[9] The chance, in generating random binary patterns, of producing one of the members of the domain X studied was very low and not taken into account, since there are 2^{8n} such binary patterns for sequences of length n, a number much greater than the size of the testing set used. There was thus a 1 in $2^{16}/26^2 = 96$ chance with $n = 2$, a 1 in 955 chance for $n = 3$, and a less than 1 in 400,000 chance for larger n's, to randomly generate a binary pattern corresponding to a pattern of the corresponding domain. Furthermore, any errors produced by ignoring this factor would be towards lower estimates of d.

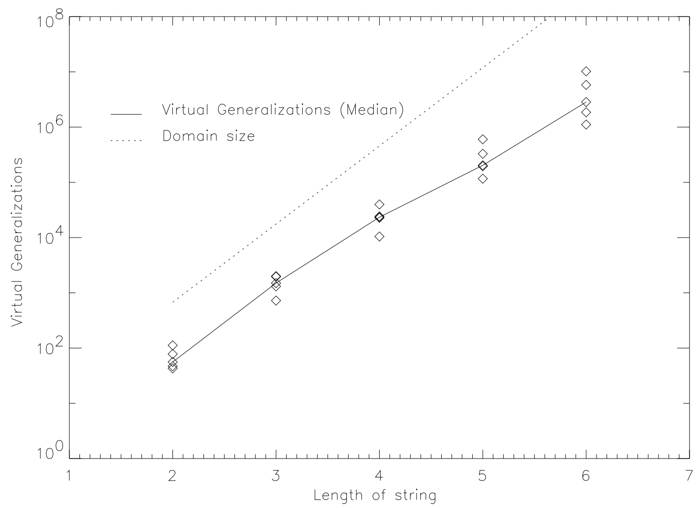


Figure 4.13: Nearly exponential growth of virtual generalizations for networks trained on sets of size 50, with $A = 26$, as n varies from 2 to 6.

generalization as a function of n, indicates that as the combinatorial complexity of the domain increases, so does discrimination: As the number of statistical regularities of the training set increases, their capture by the networks induce less acceptance of non-members.

The upper right graph, showing discrimination as a function of the alphabet size A, indicates the same trend that was observed with generalizations. It can thus be tentatively explained by a similar account: Discrimination decreases as presentation frequencies of a given letter at a given position decrease, less rapidly at first than the opposite discrimination decrease due to polynomial growth of the domain with A at first, but more rapidly afterwards.

The lower left graph, showing discrimination as a function of the number of hidden units H used, was tentatively explained when the graph showing generalization as a function of H was studied.

4.6 Virtual generalizations

4.6.1 Main results

Figure 4.13 shows the number of virtual generalizations in the case of the networks trained on 50 input patterns, for $n = 2, 3, ..., 6$, with $A = 26$ and $p = 50$. As the combinatorial complexity of the domain increases, large numbers of virtual generalizations appear. For $n = 6$, for instance, we estimate that about 3,000,000 virtual generalizations exist. The growth is approximately exponential, and the growth rate is quite close to that of the size of X itself (shown in the dotted line).

Figure 4.14 shows the number of virtual generalizations for networks trained on sets of size 50, with $n = 4$ and $A = 16, 21, 26, 31, 36$. We observe that the pattern is similar to that for generalizations, although slightly less pronounced. The same explanation involving two opposite trends can thus be hypothesized.

Figure 4.15 shows the number of virtual generalizations for networks trained on sets of size 50, with $n = 4$ $A = 26$, and H varying. Again, the pattern is similar to that obtained with generalizations. Discrimination in that case, as shown in the lower left graph of figure 4.16, follows a pattern that is almost similar, but not quite: It steadily decreases between the points corresponding to $H = 20$ and $H = 25$, and instead of increasing again, slightly decreases, until the point corresponding to $H = 28$ is reached, where it increases.

It seems, then, that virtual generalizations of patterns of the domain grow slightly slower than virtual generalizations of non-members of the domain, explaining decreased discrimination up to the point of abscissa $H = 28$. Past this point the relation is reversed.

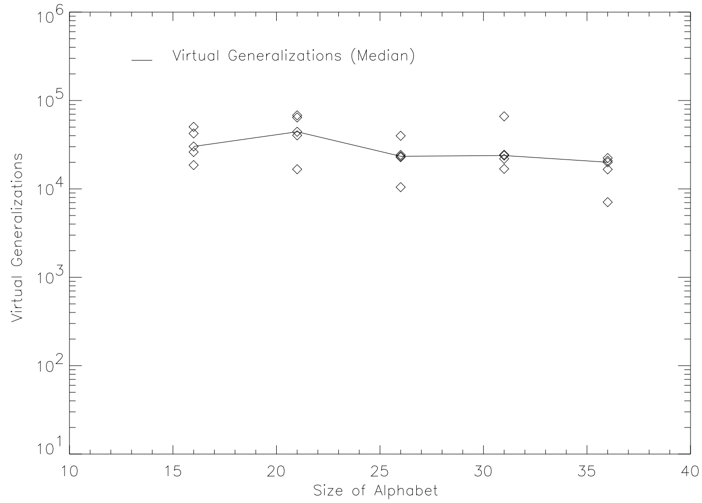


Figure 4.14: The number of virtual generalizations for networks trained on sets of sizes 50, with $n = 4$, and $A = 16, 21, 26, 31, 36$.

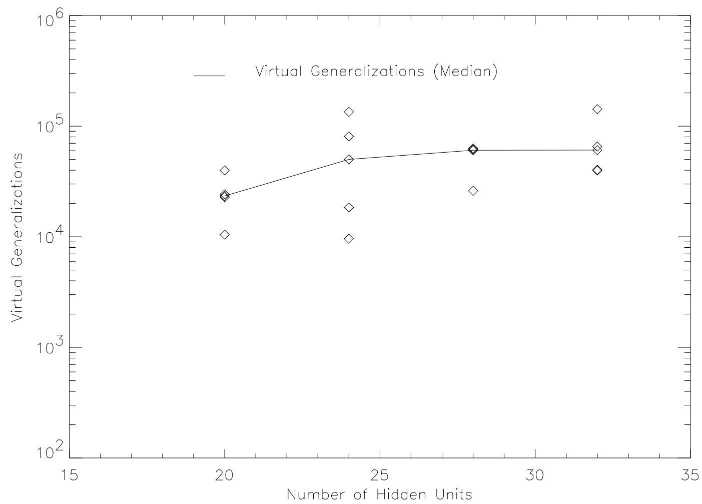


Figure 4.15: The number of virtual generalizations for networks trained on sets of sizes 50, with $n = 4$, and $A = 26$, and H varying.

Figure 4.16 shows the sum of training set sizes and virtual generalizations for networks trained on p patterns as p varies, with $n = 2$ and $A = 26$ constant. As with the case with generalizations, we show the combined sum of both training sets sizes and virtual generalizations. The figure indicates, as was the case with generalizations although here more markedly, that the number of virtual generalizations grows fast for small values of p, and slow as these values get large.

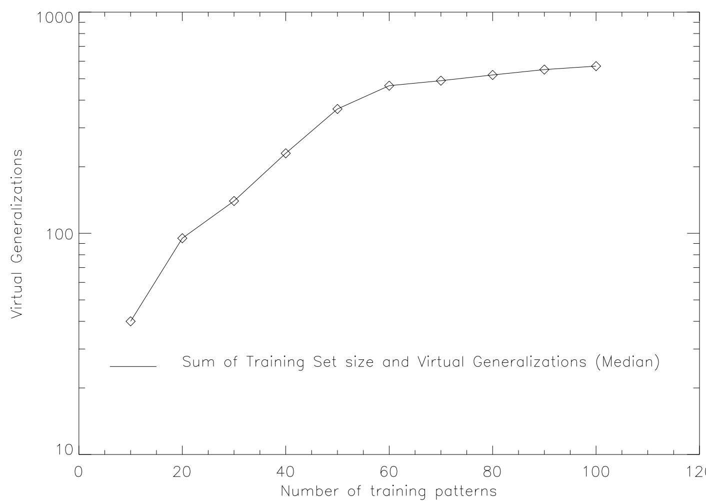


Figure 4.16: The number of virtual generalizations for networks trained on sets of sizes 50, with $n = 2$, and $A = 26$, and p varying.

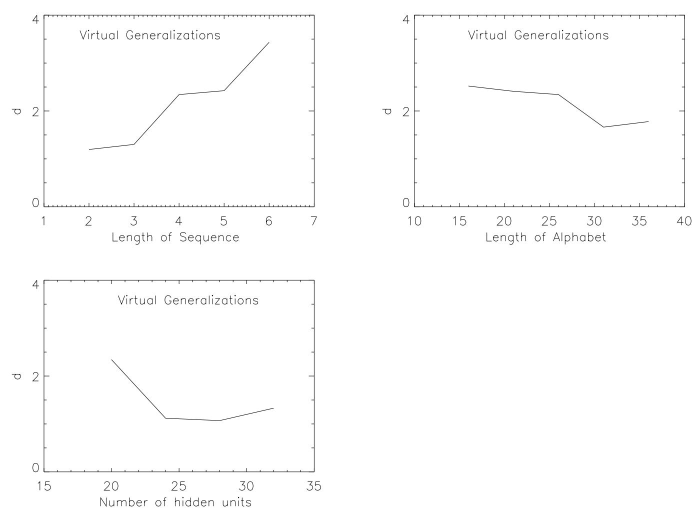


Figure 4.17: Discrimination measures for virtual generalizations

4.6.2 Discrimination tests

If the number of virtual generalizations is to give us, as was argued in chapter 2, a measure of how well the networks have induced the structure of the domains, we must also, as was the case with generalizations, measure how well the networks can discriminate between virtual generalizations of members of the domain and spurious virtual generalizations obtained for non-members of the domain. Figure 4.17 shows such measures. All trends are strikingly similar to those obtained with generalizations, except for the case of discrimination varying with hidden units, which was discussed above. We should note, however, that discrimination is under 2 for experiments done with $n = 2$ and $n = 3$, with an alphabet size of $A = 26$. This experiments, thus, loose their value if it were not for their contribution to show the overall trend in the figures.

4.6.3 Weight Updates

In all previous experiments investigating learning in the combinatorial domain $X = \prod_{i=1}^{i=n} X_i$, we allowed a potential virtual generalization to be learned with a maximum number of 5 trials. The learning rate for such learning was empirically chosen to be 20 times larger that the learning rate used for original learning of the training sets (the reader is referred to the last section of this chapter for values of these). [10] It follows, then, that all numbers previously reported, pertaining to virtual generalizations, were lower bounds: Some items, maybe, could have been learned with different learning rates or with more learning trials.

Our original intuition, however, was that since patterns of the testing sets shared in the combinatorial structure of the learned patterns, they could be learned with few learning trials with no interference, and if so, a choice of a large enough learning rate could result in a "one-trial" learning.

In the linear, no hidden units case , a new pattern can be learned in just one trial. This is because, if W is the weight matrix of a network having learned the training set, the new matrix W_{new} that has learned the pattern P with the delta learning rule with a learning rate η satisfies:

$$W_{new} P \;=\; P$$

if

$$\eta \;=\; \frac{1}{||P||^2}$$

since

$$W_{new} \;=\; W \;-\; \eta \,(WP - P)\; P^T$$

In the linear case, then, any pattern can be learned in one trial given an adequate learning rate. A pattern which was successfully learned in t trials and did not produce any interference might not, however, be learned in one trial and still produce no interference, because there is no reason to believe that the resulting matrix after t learning trials will be equal, in general, to W_{new}.

This is because, if W_t is the new matrix obtained after t learning trials of P at a learning rate η_t, t recursive applications of the delta rule

$$W_t \;=\; W \;-\; \eta \,(W_{t-1} P - P)\; P^T$$

yields

$$W_t \;=\; \left(I - \eta_t P P^T\right)^t W \;+\; f(t) P P^T$$

where I is the identity matrix and $f(t)$ is a function of t found by recursion. There is no reason to believe, one can verify, that the equation

$$W_t \;=\; W_{new} \;=\; W \;-\; \eta \,(WP - P)\; P^T$$

is verified with

[10] Except for the cases where n was 2, where that learning rate was 40 times larger.

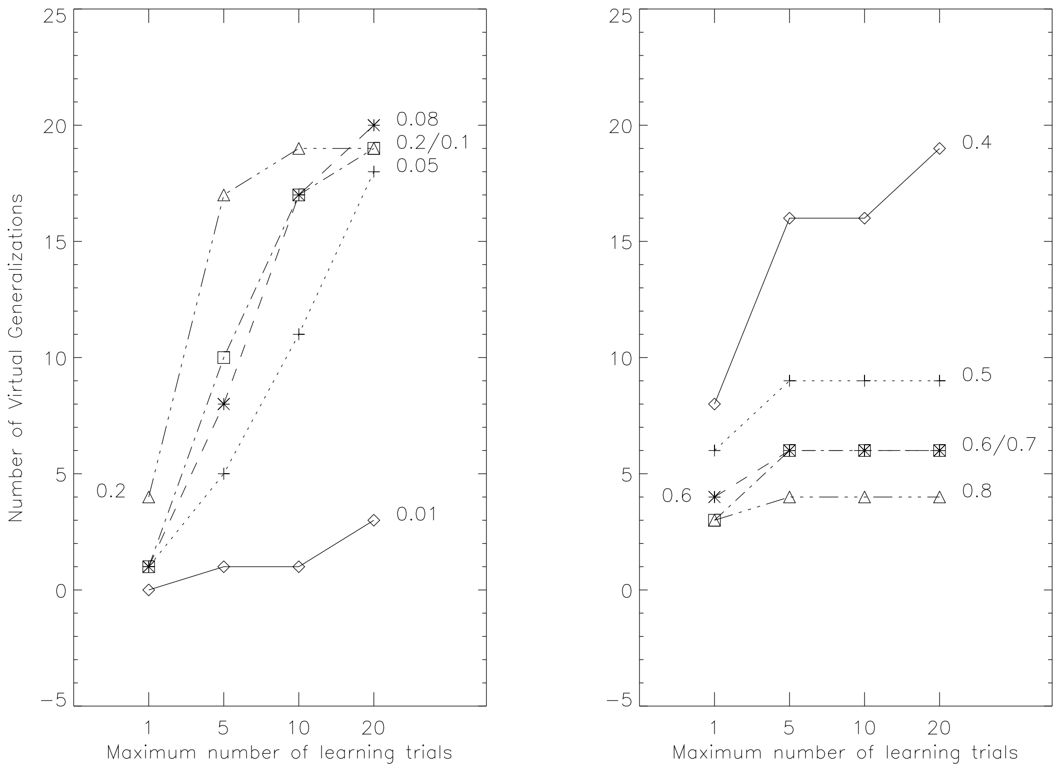


Figure 4.18: Number of virtual generalizations obtained with various learning rates and number of weight updates, in a network trained on 100 patterns, with $n = 3$, $A = 26$ and $H = 15$. The numbers adjacent to the curves are the learning rates used.

$$\eta = \frac{1}{||P||^2}$$

Within a generalization task for a structured domain, however, we hypothesize that W_t and W_{new}, although different, would both lead to little interference with the patterns of the training set correctly associated with W.

To test this hypothesis, we first conducted an experiment testing the variability of the number of virtual generalizations obtained with various learning rates and number of weight updates. A sample of 100 patterns, with $n = 3$, $A = 26$ and $H = 15$, was tested on a network having learned 50 other patterns of the same structure with a training learning rate of 0.01.

Figure 4.18 shows the number of virtual generalizations obtained with various learning rates and maximum number of weight updates. We observe rather large variability, but there is a clear and predictable trend: The number of virtual generalizations starts high and grows slowly, if at all, with the maximum number of training trials allowed for large learning rate, while it starts low but "catches up" for small learning rates, where a greater number of trials are needed. With a learning rate of 0.08, for instance, only 1 virtual generalization is found in one trial, but 20 are found if 20 learning trials our allowed. A much larger rate of 0.4, on the other hand, allows 8 virtual generalizations to be learned in one trial, 16 virtual generalizations to be learned in less that 5 trials, but only one more in less than 20 trials.

The same testing set as described above was then used to approximate the maximal number of virtual generalizations for the small problem considered. With a large number of weight updates allowed (200) and a very small learning rate (0.01), 24 virtual generalizations were found. To test how many of these could be learned in one trial, each pattern of the testing set was subjected to the learning algorithm with increasingly larger learning rates (the starting point was 0.01, the last point was 1.5, and the grain was 0.001) until the individual error on the pattern was 0. The training set was then tested for interference. The number of virtual generalizations found that way was 22. This restricted experiment, then, seems to indicate that with rather high probability, a virtual generalization can be learned in on trial, since out of an estimated 24, only 2 could not be learned in one trial.

We should also note that, although again clearly calling for further investigation, this experiment confirms our hypothesis that the number of virtual generalizations reported in all earlier experiments were estimated conservatively: With a learning rate of 0.2, figure 4.18 indicates 17 (out of an estimated 24) virtual generalizations, when the maximum number of learning trials is 5, as was the case in our experiments.

4.7 Better performance

Although we have, so far, obtained quite large numbers of generalizations and virtual generalizations, these have always been small compared to the sizes of the corresponding target domains. The networks, it can be hypothesized, had too little capacity to learn to generalize very well [11].

With alphabet sizes A of 26, they were, during training, presented with only two occurrences of a given letter at a given position, on average.

In this section, we present results pertaining to networks learning a domain $X = \prod_{i=1}^{i=n} X_i$ with a smaller alphabet, in order to confirm our hypothesis concerning the limited capacity of earlier networks. We chose a domain characterized by $n = 4$ and $A = 6$. The size of the training set was, again, $p = 50$. For a given position, a letter was thus seen, on average, about $50/6 \approx 8$ times during training.

As before, all experiments were repeated five times with different randomly chosen training sets and initial weights. Testing sets for generalizations and virtual generalizations consisted of all 1,296 patterns representing the domain X, (with the exception of patterns of the training sets) and discrimination measures were performed with five randomly selected subsets of 1,246 patterns. Since we were expecting better performance, because of the small value of A, we increased the size of the hidden layer to $H = 30$ assuming that discrimination would still be high. [12] Members of the domain were coded as before, with the exception that values of 0.1 and 0.9, instead of 0 and 1, respectively, were used. (These values were also used for random patterns used for discrimination testing). Use of such values were found to increase generalization capability, as we will see in chapter 4. Both training and testing error criterion had a value of 0.4, relative to the target activations of 0.1 or 0.9. (An output unit was thus expected to be "on the right side" of 0.5, with no margin around that value as was the case in the previously reported experiments. Since discrimination was high, as we will see, this was not a problem).

Figure 4.19 shows the number of generalizations and virtual generalizations obtained with such networks.

We observe a median number of generalizations of about 700, and a median number of virtual generalizations of about 490. The networks, then, generalize on average to about 56% of the domain, while they has been trained on 4% of the domain. Taking both true and virtual generalizations together, we find that, on average, 92% of members of the domain are either generalized or can be learned without causing any interference. For the two most performant networks, that proportion was about 98%.

With such few letters in the alphabet, the networks, also, discriminate particularly well. Figure 4.20 shows the related discrimination measures for virtual generalizations, where the same (randomly chosen) training sets and testing sets were used when testing for generalizations and virtual generalizations of members and non-members of the domain.

In our testing sets of size $6^4 = 1296$, only one virtual generalization was found when testing random binary patterns, in all of the five experiments combined. We can derive a conservative lower bound on discrimination for generalizations, assuming a "faked" one generalization in each experiment. Such a lower bound is about 6.5.

For virtual generalizations, figure 4.20 indicates that all measures are well above 2, with a median of about 4.

[11] The emphasis here is on learning, since there does exists a weight configuration, where each letter in the string is auto-associated individually without any consideration for the context in which it stands, for which the networks can perfectly auto-associate all members of the domain. This issue will be discussed at length in the next two chapters

[12] There was thus more than 6 hidden units per position, with $A = 6$. In spite of this over abundance of hidden units, generalization, as will be seen, was high.

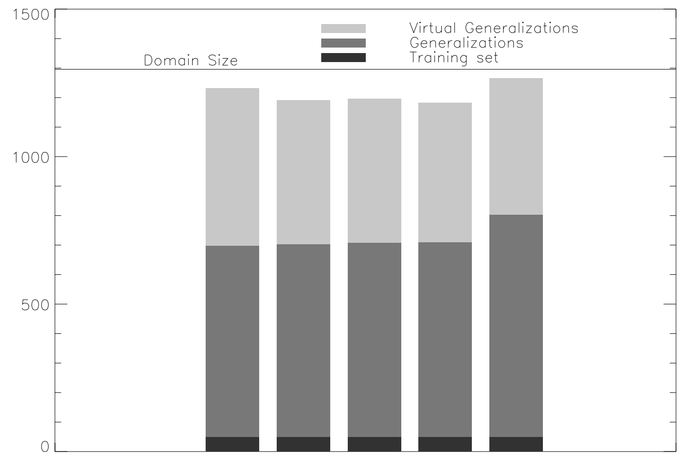


Figure 4.19: Median number of generalizations and virtual generalizations for networks trained on $p = 50$ patterns, with $n = 4$ and $A = 6$ (5 repetitions).

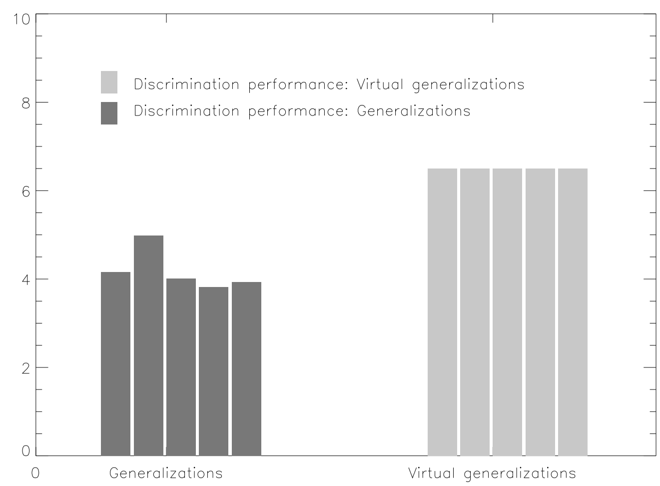


Figure 4.20: Discrimination measures with $n = 4$, $A = 6$, and $p = 50$ (Five repetitions).

4.8 Noise resistance and pattern completion capabilities

We present, in this section, experiments investigating properties that are typical of memory models: Noise resistance and patterns completion abilities.

4.8.1 Recovery from noise

In a first experiment, we tested all five networks used in the experiment reported in the last section on randomly distorted patterns of the domain. More precisely, each component of the vectors representing the members of the domain was either increased by a random value of at most 0.2 when its original value was

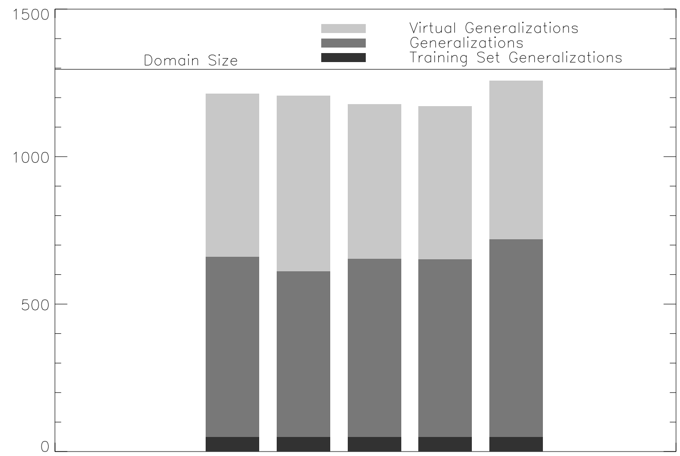


Figure 4.21: Recovery from noise

0.1, or decreased by a similar value when its original value was 0.9. The distortion, thus, always pointed towards 0.5, and changed a component, on average, by a fraction of 10% of its maximum allowed range, with a maximum change of 20%.

Figure 4.21 shows the number of true and virtual generalizations obtained, distinguishing between generalized patterns that belonged to the training set and generalized patterns that did not.

All 50 distorted patterns of the training set were correctly associated, and the number of generalizations and virtual generalizations were almost similar to those derived from undistorted patterns.

Such performance suggests that the non-linearity of the activation function is crucially used by the networks, as the weights need to learn to rely solely on whether a bit is on one side of 0.5 or on the other and as, for each hidden or output unit, the weighted sum of individual distortions of activities of all their lower adjacent units is not enough to move the sum of activities to the non-linear portion of the sigmoid. (Although 80% of the weights had a magnitude of at least 0.1, and roughly half were negative).

4.8.2 Pattern completion

The following experiments was aimed at testing pattern completion capabilities of the network. In the first one, we systematically "zeroed out" out 2 bits in the 8-bit representations of the letters of the domain. That is, we chose randomly two positive bits for each representation of a letter and flipped it to its corresponding opposite value. Each 4 letters of a pattern representing a member of the domain, then, were, with four bits on on average, half "destroyed". The target patterns were the original undistorted patterns. Figure 4.22 shows the performance of the networks on the modified patterns.

We estimate a median number of 10 patterns of the training set which could be learned with no interference, (that is, which left intact performance of the original, unmodified, training patterns), and a median number of about 200 virtual generalizations which were not from the original training set. No patterns were generalized.

Another experiment investigated how the network reacted to presentations of patterns that had one letter "zeroed out". That is, a filler representing any of the six letters in the domain was systematically replaced by the null vector in one specific position. Note that on average, fillers will have $8/2 = 4$ bits on. So "wiping out" a letter consisted in flipping, on average, 4 bits of one of the letter representations making the patterns. It should be noted that since the letters in a sequence are uncorrelated, we should expect very poor generalization if the network learns to behave in a systematic way. This is confirmed in figure 4.23, which shows the (small) number of virtual generalizations obtained. No generalizations were obtained.

We estimate a median number of about 16 patterns of the training set which could be learned with no interference, and a median number of about 288 virtual generalizations which were not from the original

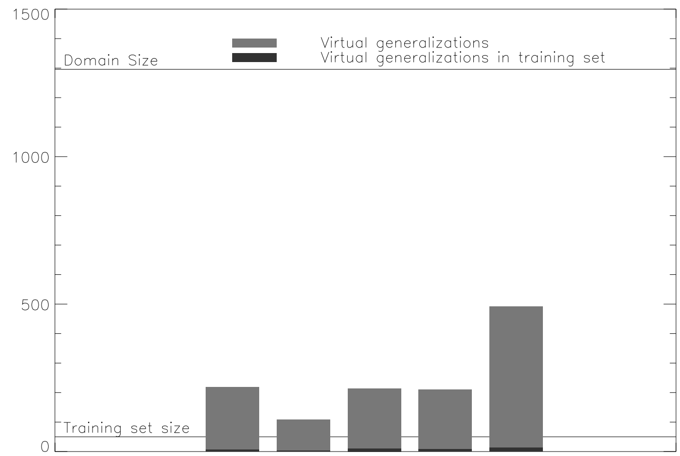


Figure 4.22: Pattern completion

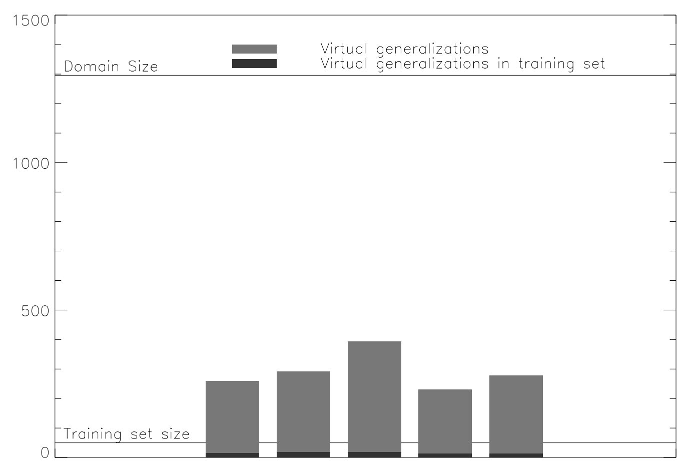


Figure 4.23: Recovering from one zeroed out letter

training set.

It is interesting to note that, although the bit distribution of the letter representations varied with the representations, the letter distributions at the "zeroed out" position, for patterns which were reconstructed through learning, was completely even. We show below the randomly-generated 8-bit representations of the letter used.

```
{1, 0, 0, 1, 0, 1, 1, 0 }  /* A */
{1, 0, 1, 0, 1, 1, 0, 0 }  /* B */
{0, 0, 1, 1, 1, 1, 1, 1 }  /* C */
{0, 0, 0, 1, 0, 0, 0, 1 }  /* D */
{1, 0, 1, 0, 0, 0, 0, 1 }  /* E */
{1, 0, 0, 0, 1, 0, 0, 0 }  /* F */
```

Although some representations, like the ones corresponding to D and F, have only 2 bits on, patterns with such letters in the zeroed out position were not predominant in the set of virtual generalizations obtained. In fact, all letters had an equal chance of figuring at the "zeroed out" position. The networks, thus, did not choose to learn to reconstruct the patterns with missing letters that were the closest, in term of incorrect bits, to the null vector corresponding to a "wipe out" position.

4.8.3 Discussion

The experiment reported above indicate that our networks fall short of presenting the kind of pattern completion ability that memory models require. Although noise resistance is rather good, the patterns of the training set were not "completed" in a significant manner. This might not be so surprising since the networks are feed-forward and thus lack recurrent connections on the input and output units. Such recurrent connections have been shown to enhance pattern completion ability with back-propagation (Almeida, 1987).

4.9 The need for non-linearity

All the networks we have used in our experiments have a non linear activation function. In this section, we report on small experiments involving learning the domains with a linear (the identity) activity function.

In a first series of simulations, we attempted to replicate, with the same networks and under the same conditions except for the use of the linear activation function, the simulation performed on the domains with $A = 26$, $p = 50$ and n =4. Five randomly chosen subsets of 50 patterns representing members of the domain were thus generated, and were trained on five networks with different random initial weights, with the same architecture that was used earlier.

It turned out that is was impossible to conclude this first series of experiments. Various learning rates were used, and large numbers of epochs were allowed (more than 5,000), but the networks could not learn the training sets to criterion, although such learning could be completed in a few hundred epochs at most by the non-linear networks.

In a second series of experiments, we repeated our approach, but this time for the much easier problem, reported earlier in the case of non-linear networks, consisting in learning the domain with $p = 50$, $n = 4$ and $A = 6$. Figure 4.24 shows the numbers of generalization and virtual memories obtained, and compares these with the corresponding quantities obtained with non-linear networks.

The linear networks perform clearly better, with an estimated median number of generalizations of about 1,060, which represents about 82 % of the number of patterns in the domain.

Figure 4.25 shows results of discrimination measures, again comparing them with the corresponding measures obtained with the non-linear networks, where the same (randomly chosen) training sets and testing sets were used when testing for generalizations and virtual generalizations members and non-members of the domain. The linear networks, clearly, discriminate much less. Judging from these results, it can be hypothesized that these linear networks "attempt" to bypass the regularities of the training sets and try to approximate the identity function. Since the size of the hidden layer, in the experiment reported here, is just slightly smaller than the size of the input and output layers ($H = 30$), the approximation is possible. It is far from perfect, but as the rather good discrimination measures of value above 2 show, it is good enough to induce very good generalization. For harder problems like those where the alphabet size A is larger and/or the size of the hidden layer is reduced, however, such an approximation is not possible any more and the linear networks, therefore, cannot learn the training sets.

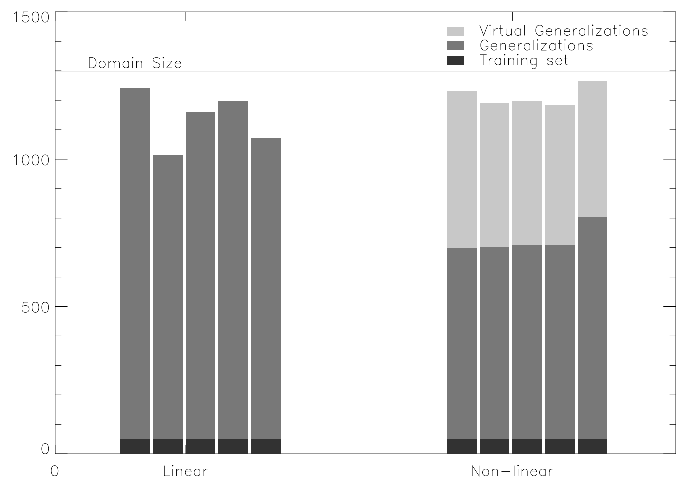


Figure 4.24: Comparison of generalization by linear and non-linear networks

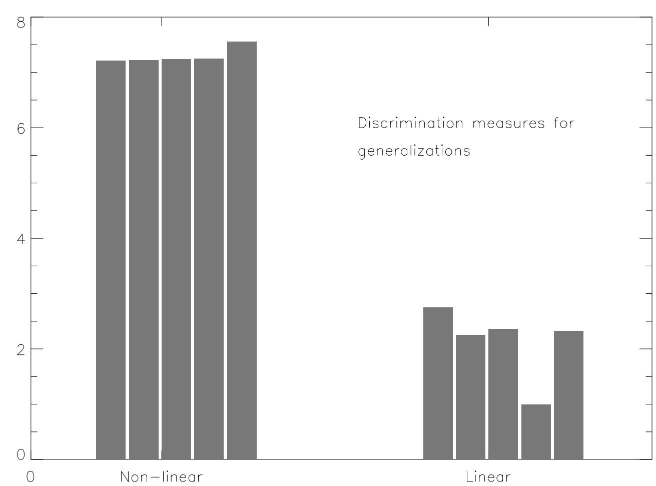


Figure 4.25: Comparison of discrimination for linear and non-linear networks, learning the same task

4.10 Varying the error criterion in training and testing

4.10.1 Generalizations and virtual generalizations

In all our previous experiments, we have used a constant error criterion, (here and henceforth ϵ) of 0.4 for training, and an error criterion for testing (here and henceforth ϵ_t) of the same value.

The following simulations were aimed at estimating the dependence of the number of generalizations and virtual generalizations on these two quantities. All the following experiments were conducted with $A = 26$, $H = 20$, $p = 50$ and $n = 4$, testing being performed 5 times with different initial random weights and training sets. 10,000 random binary patterns and 10,000 random patterns of the domain were used for testing. It should be mentioned that in this experiment, for better comparison purposes, the same (randomly chosen) training sets and testing sets were used when testing for generalizations and virtual generalizations of members

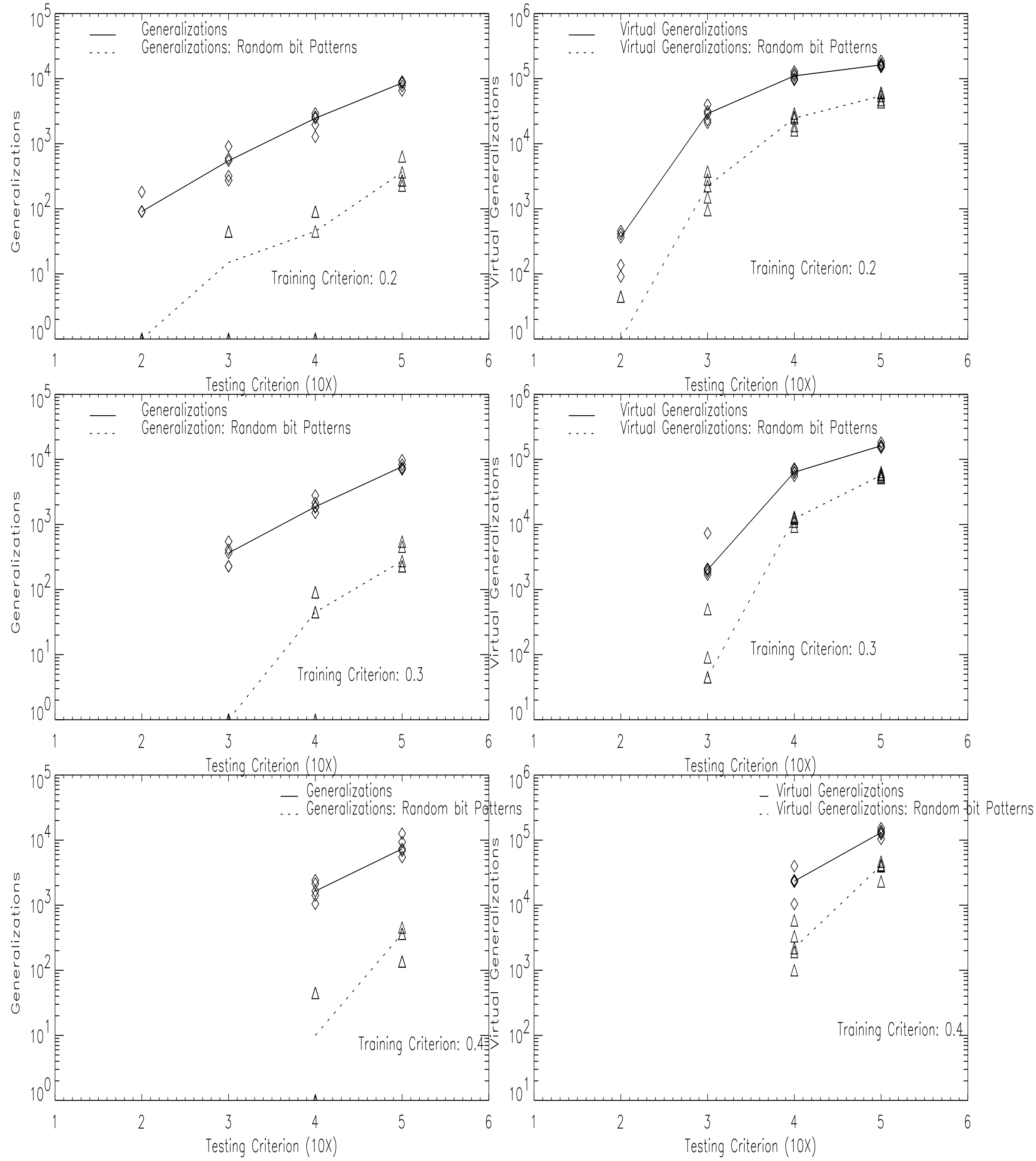


Figure 4.26: Generalizations and virtual generalizations obtained with various values of ϵ and ϵ_t, for networks learning the domain with $n = 4$, $A = 26$, $p = 50$ and $H = 20$

and non-members of the domain.

Figure 4.26 combines semi-logarithmic graphs obtained with various values of ϵ and ϵ_t. The dependence of the numbers of generalizations and "spurious" random binary pattern generalizations on the testing error criterion ϵ_t, obtained with different values of the training error ϵ, are shown on graphs appearing on the left side of the figure, while the corresponding numbers for virtual generalizations are shown on the right side. The relative positions of both the generalization and virtual generalizations curves for the various values of ϵ and ϵ_t, will be shown in a separate figure.

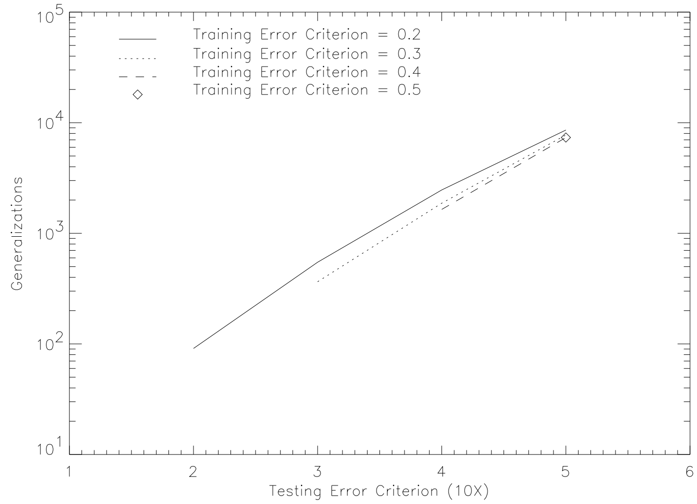


Figure 4.27: Generalizations: Comparison with training epsilon and testing epsilon varying.

The graphs displayed on the left side of the figure suggest an approximately exponential growth of both spurious and domain generalizations as the training error criterion ϵ grows, for all values of the testing error criterion ϵ_t. The rather constant height between the approximately parallel curves corresponding to domain and spurious generalizations also suggest that no matter how harsh the training error criterion is, the number of spurious generalizations is always almost constantly proportional to the number of domain generalizations. That proportionality constant is about two orders of magnitude, which loosely corresponds to a discrimination measures of 2.

Graphs displayed on the right side of the figure, pertaining to spurious and domain virtual generalizations, suggest that these grow rather less exponentially with the training error criterion ϵ. We observe, again, a rather constant height between each pair of curves corresponding to spurious and domain virtual generalizations, that similarly suggests that spurious virtual generalizations are almost constantly fewer than domain virtual generalizations, by about one order of magnitude.

Figure 4.27 shows how the various domain generalization curves as a function of the testing error criterion ϵ_t, obtained with various values of the training error criterion ϵ, compare. The graph suggests an only slight loss of generalization power, estimated at a maximum of one tenth of an order of magnitude, for a given testing error criterion, when training error criteria are decreased. High precision learning of the training set does not seem, then, to be crucial in the networks' generalizing performance. So, for example, if one will test with a testing error criterion of $\epsilon_t = 0.5$, there is no use in training with a harsh 0.2 error criterion, which will slow down (sometimes dramatically) the training process, if one is only concerned with generalization power.

Figure 4.28 shows how the corresponding virtual generalization curves compare.

In this case we note that the loss of virtual generalizations due to the use of a "harsh" training error criterion ϵ, for a given testing error criterion ϵ_t, is more severe, but seems to diminish as larger ϵ_t are used. This loss is about an order of magnitude for $\epsilon_t = 3$, half of an order of magnitude for $\epsilon_t = 4$, and less that a tenth of an order of magnitude for $\epsilon_t = 5$.

It should be mentioned that, assuming correct generalization of behavior across various domains, the graphs above indicate that by our choice of error criteria ϵ and ϵ_t of 0.4 in our previous main experiments, we have lost about one order of magnitude of virtual generalizations, but hardly any generalizations, by chosing a testing error criterion of 0.4 instead of 0.5.

4.10.2 Discrimination factors

We study, in this section, the dependence of discrimination measures on both ϵ and ϵ_t. Figure 4.29 shows discrimination measures when ϵ and ϵ_t vary, for generalizations. Due to the parallelism of the spurious and

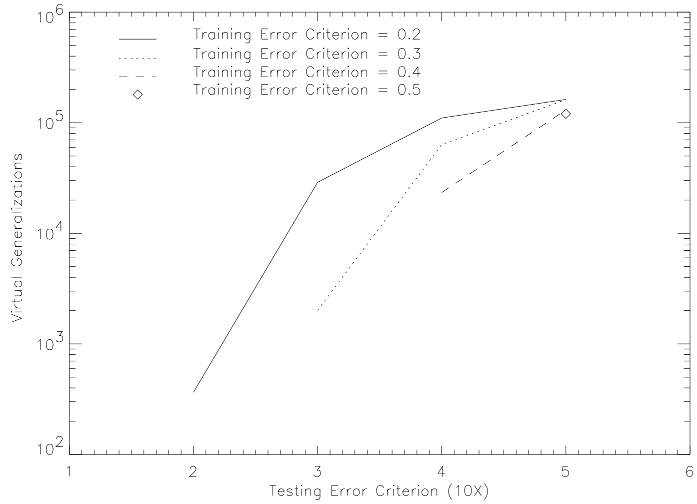


Figure 4.28: Virtual Generalizations: Comparison with training epsilon and testing epsilon varying.

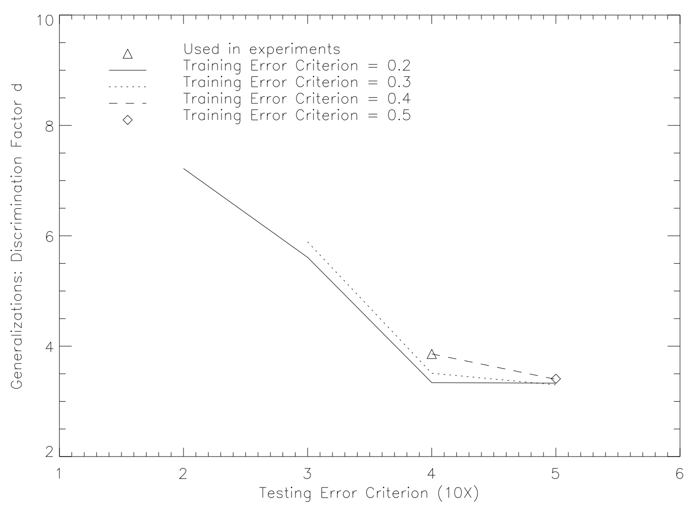


Figure 4.29: Generalizations : Discrimination factor d for various values of training and testing epsilon .

domain generalization curves seen earlier, the graph is closely patterned after the graph shown in figure 4.27, the trend being reversed. This similarity can simply be explained by the fact that when P_X, the probability that a member of X is a generalization, and $P_{\bar{X}}$, the probability that a non-member of X is a generalization, are small, the discrimination measure d close to $\log(P_X/P_{\bar{X}})$ As was the case with generalization power, discrimination power, for a given testing error criterion ϵ_t, is not much different across various training error criteria ϵ. Figure 4.30 shows discrimination measures when training and testing error criteria ϵ and ϵ_t vary, for virtual generalizations. Again, and for the same reason, the graph is closely patterned after the corresponding graph shown in figure 4.28, the trend being reversed. Discrimination power, for a given testing error criterion ϵ_t, varies more across various training error criteria ϵ, and it is critical here since the discrimination values are around 2.

The choice of a value of 0.4 for testing and training error criteria in our main experiments seems to be

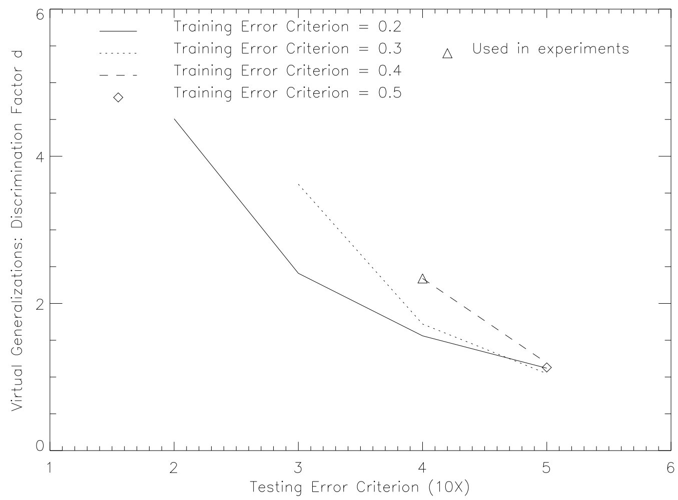


Figure 4.30: Virtual Generalizations: Discrimination factor d for various values of training and testing epsilon

close to an optimal choice: we assured a discrimination measure above 2 (a reasonable value in most human performance psychological experiments) while maximizing the number of virtual generalizations. As the graph shows, a training and/or testing error criterion of 0.5, while improving generalization and virtual generalization ability, would not have yielded satisfactory discrimination.

4.11 Discussion

The experiments conducted in this chapter suggest that a connectionist network of the most generic nature is capable of learning to extract, from a small number of compositional representations coding instances of a combinatorial domain, the systematic regularity of such a domain. Such extraction leads to generative behavior, as exponential growth of both true and virtual generalizations is observed when the complexity of the domain increases.

4.12 Experimental parameters

We review, in this section, relevant parameters pertaining to the simulations reported in this chapter.

In all experiments, the momentum was 0.9 and the error criterion ϵ was 0.4 both for training and for testing of virtual generalizations, unless it was explicitly stated. The learning rate was 0.01 for training and 0.2 for virtual generalization learning, except for $n = 2$ where the training learning rate was 0.005.

The vectors representing members of the sets X_i's were random binary vectors of length 8. H, the number of hidden units, was linearly increased as n increased according to $H = 5n$, resulting in a constant compression factor of 8:5 from input to hidden units. Initial weights were generated pseudo-randomly, with equal probability in the interval [-0.5, 0.5].

Patterns in the training set were presented to the network in a random order during an epoch, and weights were updated after each pattern. Because gradient descent suffers from the problem of local minima, some training sets could not be learned in a reasonable time. When that happened we simply started the experiment over. When the error for all examples of the training sets reached 0, we trained again for 10 epochs to ensure stability. (Since weights are changed after each pattern presentation, a total error of 0 at the end of one epoch does not guarantee that the next will still contain error-free patterns.)

All networks were standard three-layer feed-forward back-propagation networks, with bias on all hidden

Table 4.2: Experimental parameters

Figure	X: A	X: n	X: Constraint	X: H	X: p	Epochs
4.2	26	4	English	20	100	255
4.3	26	4	English	20	100	255
4.4	26	4	English	20	100	555
4.5	26	Varies	none	Varies (5n)	50	119-486
4.7	Varies	4	none	20	50	164-259
4.9	Varies	4	none	20	50	164-259
4.10	26	4	none	10	Varies	15-1256
4.12	Varies	Varies	none	Varies	Varies	15-1256
4.13	26	Varies	none	Varies (5n)	50	119-486
4.14	Varies	4	none	20	50	164-259
4.15	Varies	4	none	20	50	164-259
4.16	26	4	none	10	Varies	15-1256
4.17	Varies	Varies	none	Varies	Varies	15-1256
4.18	26	3	none	15	100	264-385
4.19	4	6	none	30	50	25-41
4.20	4	6	none	30	50	25-41
4.21	4	6	none	30	50	25-41
4.22	4	6	none	30	50	25-41
4.23	4	6	none	30	50	25-41
4.24	4	6	none	30	50	20-29
4.25	4	6	none	30	50	20-29
4.26	Varies	4	none	20	50	*
4.27	Varies	4	none	20	50	*
4.28	Varies	4	none	20	50	*
4.30	Varies	4	none	20	50	*
4.29	Varies	4	none	20	50	*

and output units. Computer simulations were performed with the back propagation simulator of (McClelland and Rumelhart, 1988), where source code modifications allowed generation of the relevant measures.

Table 4.2 summarizes relevant parameters for our simulations. We only show minima and maxima for the number of epochs during training displayed in the rightmost column.

Chapter 5

Distributed representations

5.1 Introduction

The connectionist representations that were used in all experiments reported in the last chapter were distributed: For a given network, an input unit participated in the coding of many different members of the domain being learned. For a given structured concept of the domain being learned, however, the tensor product representation scheme used coded each constituent of the member on a dedicated pool of units. Using the terminology of chapter 2, then, the representations used were semi-distributed: While the constituents, the fillers $x_i \in X_i$, were represented in a distributed manner, the roles these constituents played within the structured member of the domain they belonged to were represented locally.

It could be argued that the performance, in terms of systematicity and generativity, of the networks presented earlier was precisely due to our use of local role representations. Since all constituents with a given role were coded on a unique and distinct pool of units, one could indeed conjecture that the networks learned to independently associate each constituent with itself, without any consideration for the others, and it could do so solely because the role representations were local. Such independent associations would require that the networks have learned to dedicate pools of hidden units for each constituent. The weights of the networks, then, would exhibit a vertical decomposition: For a given constituent, connections from its dedicated pool of hidden units to other constituents would be null.

It is important to note at this point that such learning, if it is the kind of learning that occurred, is far from trivial [1]. It requires the discovery of the statistical regularity of the training sets by the networks which, because of the full connectivity between two layers, are insensitive to permutations of input units as far as learning and performance are concerned. These networks are thus "unaware" of the dedication of pools of units to constituents of a given role that is induced by the particular distributed representation used, and have to learn it.

Rephrasing the above using the connectionist construction of concepts theory terminology, such learning would require that the networks have constructed a perfect cognitive map, (set of weights in this case) eliminating the perspective dependence of the patterns of the training set. That is, the network weights would have learned to process each pattern of the training set as a particular instance of an n-letter sequence, instead of a perspective dependent random collection of bits devoid of any regularity.

Such a conclusion, that the performance obtained depended on the use of local role representations, will not follow, however, as we will show in the next section that the networks could not have possibly **exactly** auto-associated each letter individually. We will, furthermore, make the point that for all the networks using semi-distributed representations that we have used, there exists an equivalent network using fully-distributed representations that would perform in exactly the same way. The use of local role representations, thus, was not crucial for our experiments, as long as the weights of such an equivalent network can be learned. In the last part of this chapter we will report on experiments suggesting that this is the case, as we will observe that some networks using fully distributed representations can learn to perform as well as networks using semi-distributed representations.

[1] The next chapter will report on experiments where the networks learn this.

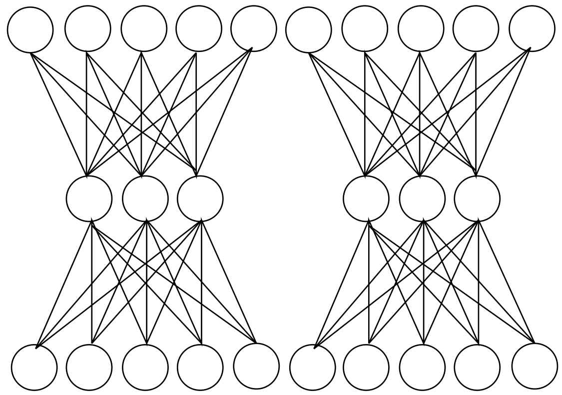


Figure 5.1: Example of a network architecture which would allow for the independent auto-association of individual letters of two-letter sequences, where each letter would be coded over 5 units.

5.2 Vertical decomposition

Let us suppose that our networks have recognized that the training patterns ("strings") are all composed of n subpatterns ("letters") that are representations of members of the independent sets X_i's, and that they have learned to auto-associate, for each training pattern, each subpattern of a given set X_i **independently** of the subpatterns of the other sets. Figure 5.1 shows a network architecture which would accomplish this for $n = 2$, if 5 units were used to code each letter in 2-letter sequences.

One could infer, then, the following two propositions:

1. The networks will necessarily perform perfectly on all untrained patterns of the domain, as long as each "letter", for each position, was presented at least once during training. This is because the network, having learned to independently auto-associate every letter for a given position, will correctly associate any untrained string containing these letters.

2. The networks will necessarily have learned to "dedicate" hidden units for each letter of strings of the domain, and cross-connections between hidden units dedicated to one letter and input units coding another letter will be non existing. If it was not so, the networks, via hidden units responding to two or more letters, would not auto-associate each letter independently.

3. When the number n of constituents does not evenly divide the number of hidden units used, a network cannot, possibly, exhibit exact vertical weights decomposition, unless pools of hidden units with variable sizes are discovered by the networks to auto-associate each constituent at a given location. Since each constituent plays, in the domain we have chosen, a symmetric role with respect to the symbolic structure it belongs to, it seems highly unlikely that such a variable size hidden unit spooling could be discovered [2]. It follows, then, that if weights vertical decomposition corresponded to the target mapping, the number of hidden units used would be crucial for the performance of the networks. But as experiments reported in chapter 4 have suggested, this is not the case: the relatively weak sensitivity to the number of hidden units in terms of performance suggests the presence of highly distributed representations on the hidden layers, rather than local. This observation concurs with the previous arguments according to which exact weights decomposition could not possibly have occurred.

These two properties were not satisfied in our experiments: No network exhibited perfect performance on the untrained members of the domain, nor presented vertical decomposition in the weights. Figure 5.2, for instance, shows the weights developed by one of the networks reported in the previous chapter, with $n = 4$, $A = 26$, $p = 50$, and $H = 20$. The error criterion used was $\epsilon = 0.4$. No vertical decomposition, as we can see, occurs. For comparison, an example of such a decomposition is shown in figure 5.3. Such weights were

[2] It could be said that some hidden units could spontaneously turn themselves off to allow for even sized pools. This was not observed, however.

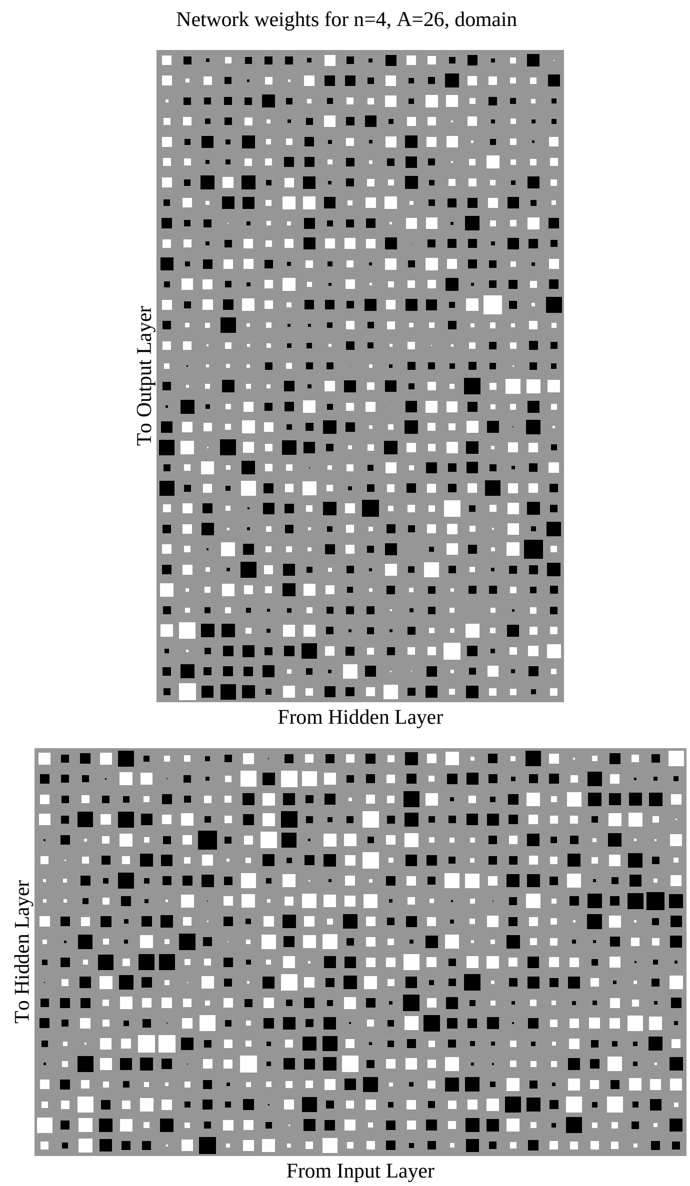


Figure 5.2: The weights of a network having learned the domain with $n = 4$, $A = 26$, and $p = 50$, with 20 hidden units. No vertical decomposition is obvious

obtained when the technique of weights eliminations was used during training. A full discussion of this case appears in in chapter 6.

We could object, here, that since the training set was randomly generated and since each letter, for a given position, was presented only twice on average, a possible lack of presentation of a letter for a given position might be responsible for the absence of vertical decomposition. Figure 5.4 addresses this objection, as it shows the weights developed by a network learning the domain with $n = 4$, $A = 6$, and $H = 20$. All letters, at a given possible position, appeared in the training set. The error criterion used was $\epsilon = 0.4$. Again, no vertical decomposition can be observed. We could still make the point that, since a loose training error criterion ϵ of 0.4 was used, and since out networks never achieve perfect generalization performance, training stopped before the weights could possibly show any vertical decomposition. Furthermore, it could be argued that the only way a network could possibly perfectly generalize to all untrained members of the

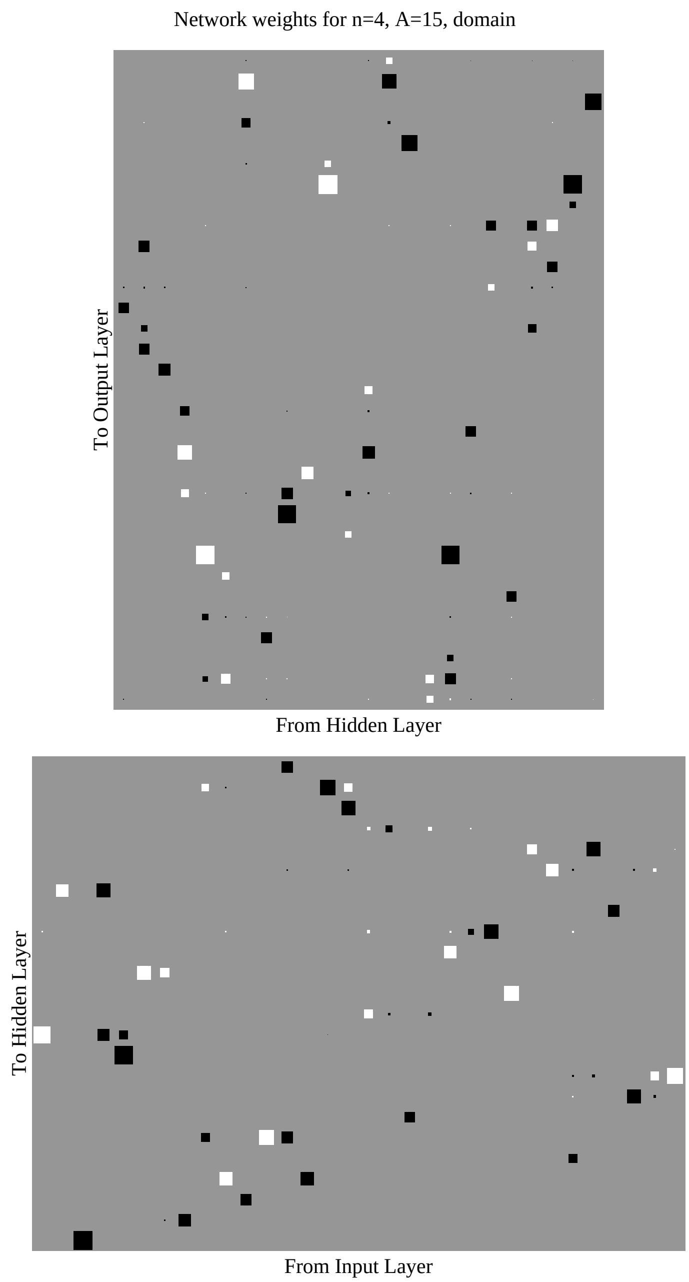


Figure 5.3: Example of a network exhibiting vertical weights decomposition, in the case $n = 4$ and $A = 15$.

domain would be to discover weights exhibiting "perfect" weight decomposition, thus exactly implementing a symbolic system. This could well be true, although figure 5.5, which shows the weights of the same network used in the previous figure, although this time a strict error criterion $\epsilon = 0.05$ is used, displays weights that are surprisingly similarly unstructured. [3] The error criterion used was the smallest possible one which allowed

[3] A simple glance at the weights of the network can assure us of the absence of weights decomposition, but cannot convince us that an approximation of such a decomposition has not been computed, even if the weights do not seem to present any structure: Weights at a given layer corresponding to a vertically decomposed solution could be transformed by a matrix approximating the identity matrix in such a way that the resulting solution would not be far from a vertically decomposed solution, and yet look completely unstructured. Our observations, confirmed by others ((Sirat, 1990), (McMillan, 1991)) have been that weights exhibiting structure are almost never seen. Statistical techniques as those used in chapter 7, however, can allow to entangle a possibly hidden structure.

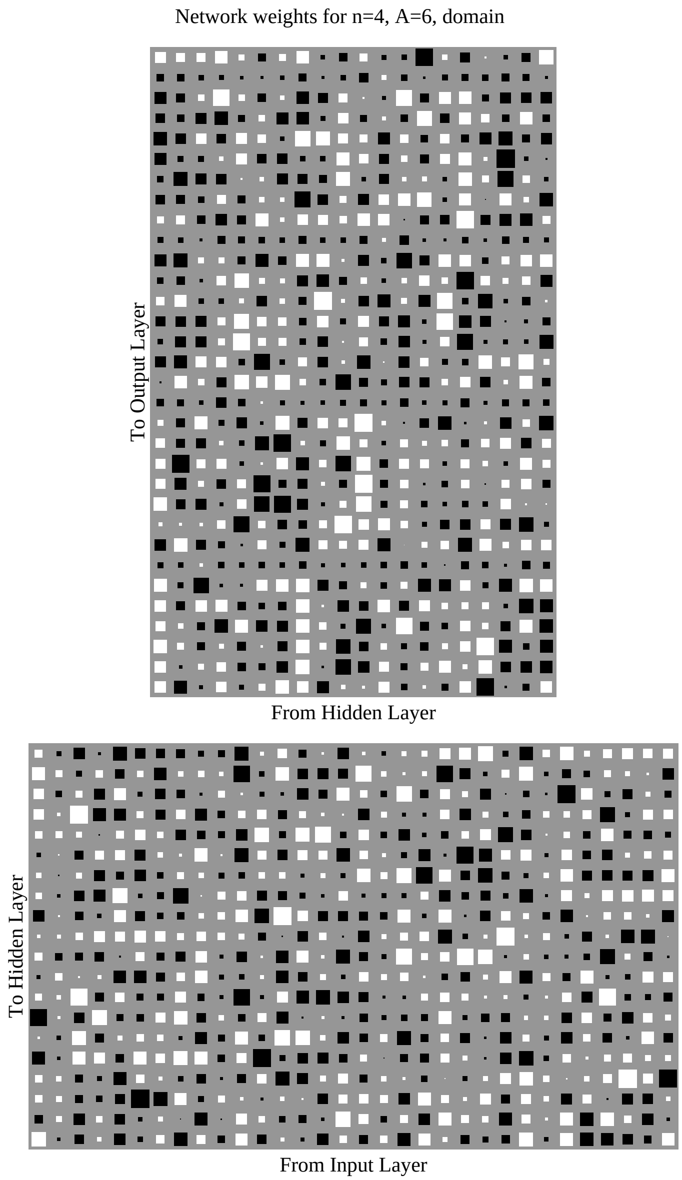


Figure 5.4: The weights of a network having learned all patterns, with an error criterion $\epsilon = 0.1$, of a domain with $n = 4$, $A = 6$, and $H = 20$. The error criterion used was $\epsilon = 0.4$ No vertical decomposition, again, is observed

learning of the 50 patterns of the training set. Figure 5.6, however, shows the weights of a network having learned all 16 patterns of a domain corresponding to $n = 4$, and $A = 2$, with an equally strict error criterion ϵ of 0.05. For this trivial problem, then, vertical decomposition, although far from Perfect, now clearly appears.

In another experiment, we constrained a network to perform weight vertical decomposition, by dividing the set of hidden units into n groups of units, connecting each group with a unique pool of input and output units coding letters for a given position. The architecture of such a network, presented previously in figure 5.1, is shown again here for clarity in figure 5.7, for $n = 2$. With such an architecture, we first tried to replicate an experiment performed in chapter 4, where the domain used corresponded to $n = 2$ and $A = 26$. The hidden layer had 10 units, and was thus divided in 2 groups of 5 units. The training set of size 50 was semi-randomly chosen, since we imposed the constraint that each of the 26 letters had to appear at least once in each position. This constraint was satisfied by successively and randomly generating training sets, and keeping the first one for which each letter of the alphabet appeared at least once for any of the two positions.

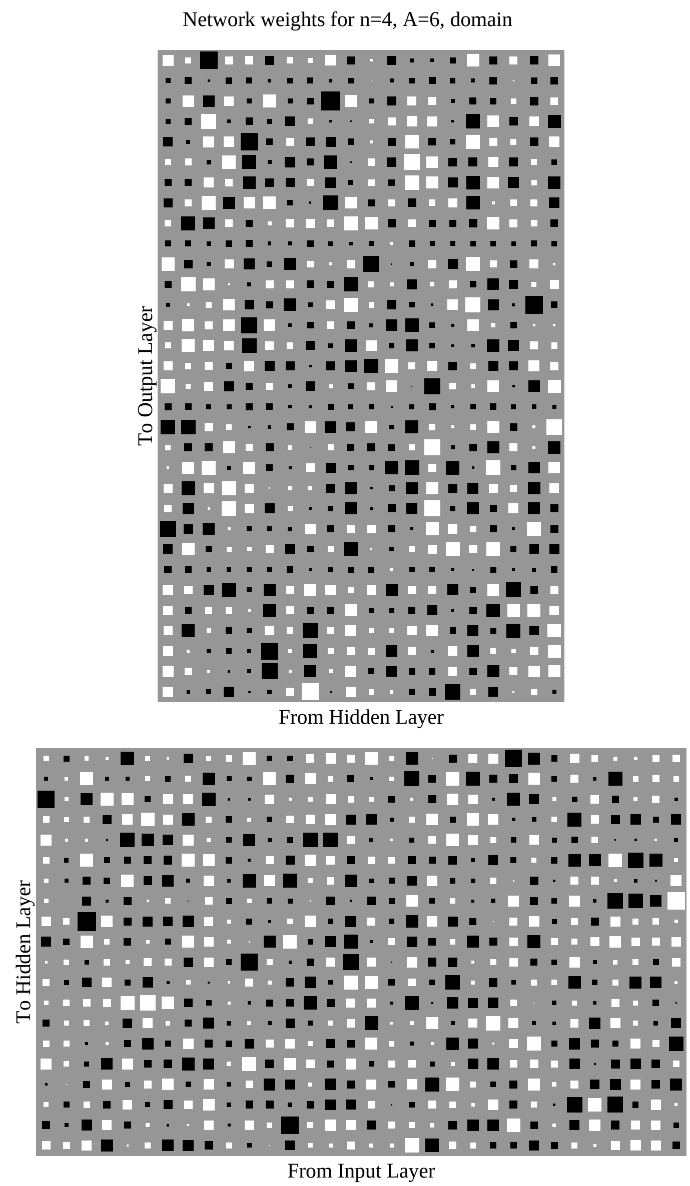


Figure 5.5: The weights of a network 50 patterns, with an error criterion $\epsilon = 0.01$, of a domain with $n = 4$, $A = 6$, and $H = 20$. There seems to be a "hint" of vertical decomposition.

It turned out that such a training set could not be learned by the "severed" network: 5 hidden units were not enough for proper auto-association of 26 different letters. The equivalent "full" networks, thus, needed extra units to learn the training set. When A was reduced from 26 to 20, the training set could be auto-associated to criterion and the network, trivially, generalized perfectly on all 400 members of the domain. Figure 5.8 shows such perfect performance, and compares the number of generalizations and virtual generalizations obtained with the "severed" network and an equivalent "full" network, for the same training and testing conditions. Figure 5.9 shows the vertically decomposed weights of the constrained network.

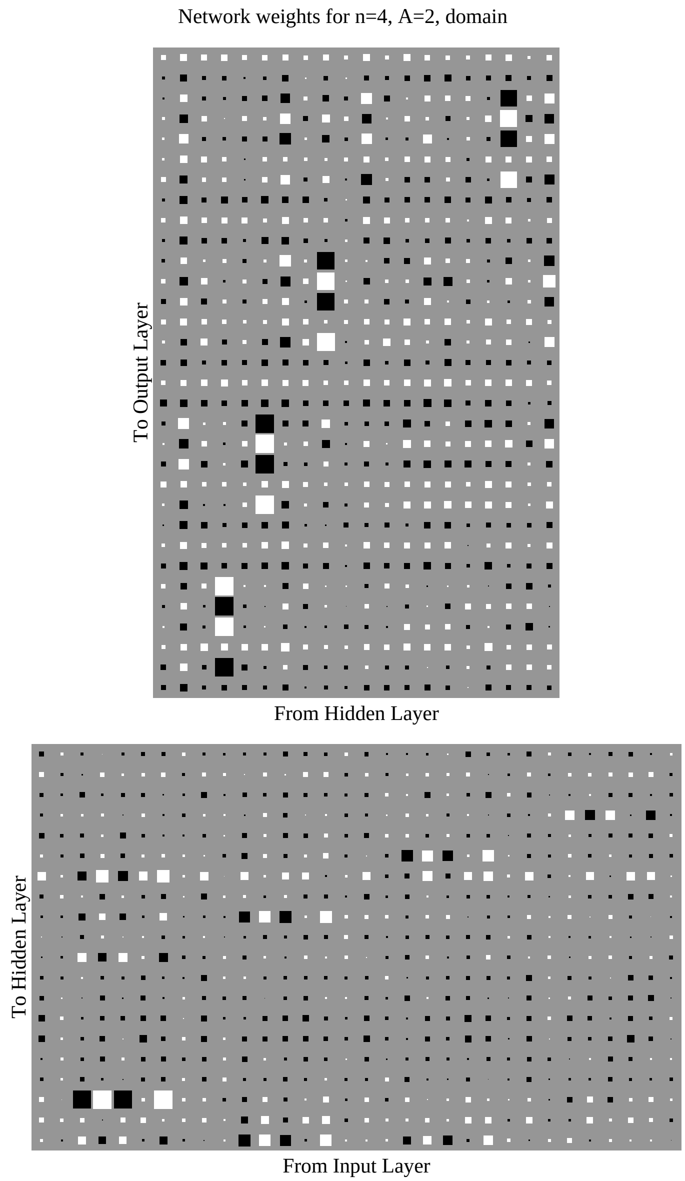


Figure 5.6: The weights of a network having learned all patterns, with an error criterion $\epsilon = 0.05$, of a domain with $n = 4$, $A = 2$, and $H = 20$. Vertical decomposition is better seen and seems to be approximated

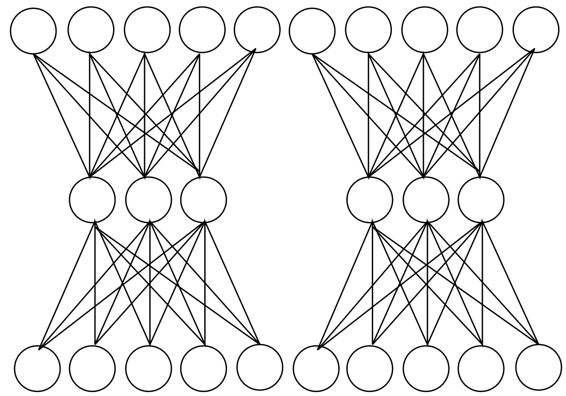


Figure 5.7: Severed network architecture

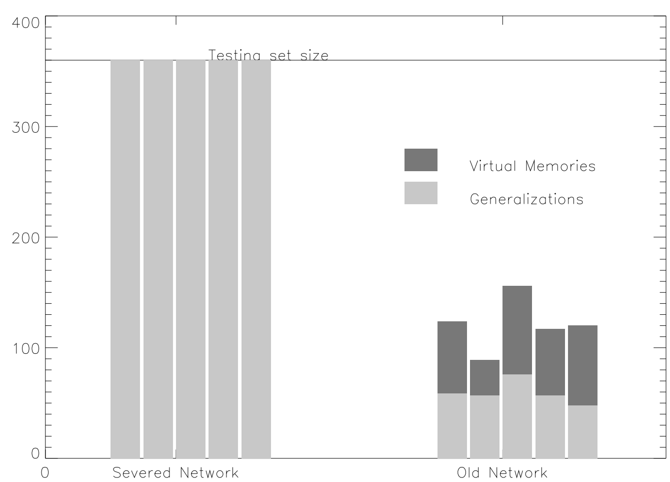


Figure 5.8: Comparison of the number of virtual generalizations and generalizations obtained with a "severed" network constrained for weights decompositions, and a full network, for a domain with $n = 2$, $A = 20$, $H = 10$ and $p = 40$, where all letters at a given position appeared in the training set. The constrained network generalizes perfectly on all patterns of the domain.

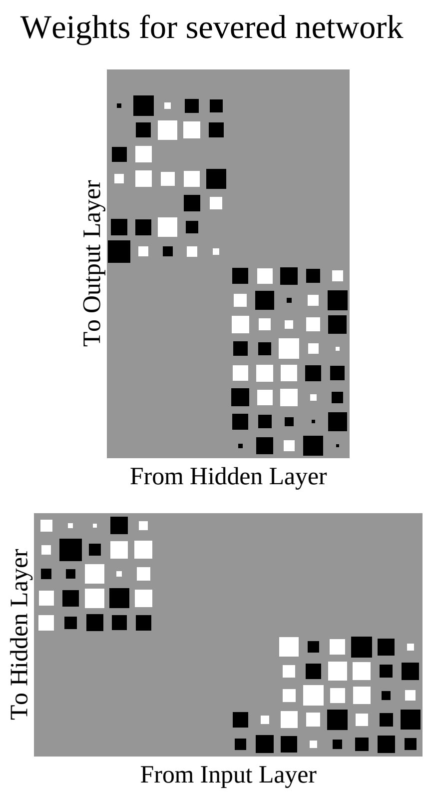


Figure 5.9: The vertically decomposed weights of a network having learned the domain with $n = 2$, $A = 20$, and $p = 40$, with 10 hidden units.

5.3 Semi-distributed and fully-distributed representations

There are a number of reasons why distributed representations are important within the connectionist paradigm. First, because they allow units to participate in the coding of many representations, they allow for fewer units and increased parallelism. Second, networks using them are much more resistant to local damage, a property which is characteristic of a wide range of cognitive processes. Finally, distributed representations can be, as was mentioned in chapter 1 and formulated by Van Gelder (1990), non-concatenative representations, and they can be non-isomorphic to the kind of concatenative representations used in symbolic systems.

5.3.1 Equivalent Networks

The distributed representations that were used in all previously reported experiments were semi-distributed, since the role representations used were local. This choice of semi-distributed representations, however, should not be crucial, since there exists, for each network computing any function of semi-distributed representations, an equivalent network that computes the same function and uses fully-distributed representations. We show the existence of such equivalent networks in the remainder of this section.

If R_1 is a distributed tensor product representational scheme mapping the p members C of X to their respective connectionist representations $R_1(C) = D_1$, then, denoting the set of constituents representations, or fillers, used $F_1 = \{f_i\}_{i=1,\ldots,n}$ and the set of roles representation used $R_1 = \{r_{1i}\}_{i=1,\ldots,n}$,

$$\forall C \in X, \quad R_1(C) \;=\; D_1 \;=\; \sum_{i=1}^{n} f_i \otimes r_{1_i}$$

If R_2 is another distributed tensor product representational scheme coding members of X, using the same fillers but different roles of the set $R_2 = \{r_{2i}\}_{i=1,\ldots,n}$, then,

$$\forall C \in X, \quad R_2(C) \;=\; D_2 \;=\; \sum_{i=1}^{n} f_i \otimes r_{2_i}$$

If the role vectors in both R_1 and R_2 are linearly independent, and they span the same vector space, then there exists an invertible linear transformation $\mathcal{M}$ that maps each vector r_{1i} of R_1 to each vector $r_{2i} = \mathcal{M}(r_{1i})$ of R_2. It follows, then, that:

$$\forall C \in X, \quad R_2(C) \;=\; D_2 \;=\; \sum_{i=1}^{n} f_i \otimes r_{2_i}$$

$$= \sum_{i=1}^{n} f_i \otimes \mathcal{M}(r_{1_i})$$

$R_2(C)$ is linear in $\sum_{i=1}^{n} f_i \otimes r_{1_i}$. It thus follows that there exists a matrix M such that

$$R_2(C) \;=\; M(\sum_{i=1}^{n} f_i \otimes r_{1_i})$$

$$= M(R_1(C))$$

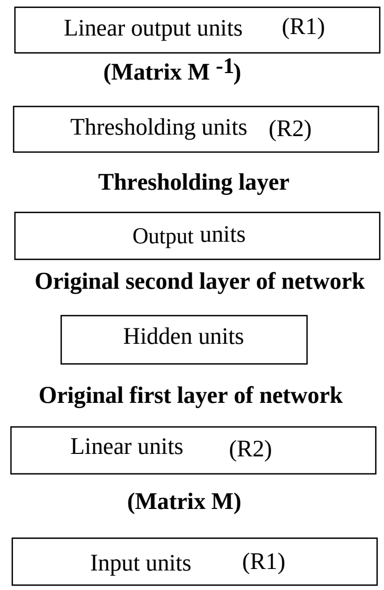


Figure 5.10: Architecture of a network associating patterns represented by a distributed tensor product scheme R_2, and computing the same boolean function as a previous network where patterns where represented by a tensor product scheme R_1 .The thresholding layer is made of one-to-one connections.

Similarly, since $\mathcal{M}$ is invertible,(it can be show that the inverse transformation from $R_1(C)$ to $R_2(C)$ is indeed M^{-1},

$$\forall C \in X, \quad R_1(C) = M^{-1}(R_2(C))$$

The two linear transformations M and M^{-1} can be implemented by two networks of linear units, where each activation function is the identity function and the weights w_{ji} on connections from a unit i to a unit j, for each network, are equal to the elements M_{ij} and M_{ij}^{-1} of the matrices associated with M and M^{-1}, respectively.

Since we used linearly independent (in the examples of chapter 4, in fact, orthonormal)roles , it follows that for any network we have used, there exists an equivalent network, using fully distributed representations (where the roles are linearly independent) [4] and computing exactly the same boolean function: If R_2 is the original semi-distributed tensor product representational scheme used, and R_1 is a fully distributed tensor product representational scheme, then an equivalent network can be constructed by:

- Connecting to the input layer of the previous network the output of the linear network computing M.
- Connecting to the output layer of the previous network a row of linear threshold units thresholding at 0.5.
- Connecting to the outputs of these linear threshold units the input units of the network computing M^{-1}.

The architecture of such equivalent networks is shown in figure 5.10.

It is important to note that when fully distributed representations are used, the original representations of the constituents of X corresponding to the symbolic letters are not "concatenated" on the units used to

[4] Since each constituent plays a symmetric role with respect to the symbolic structure it belongs too, it makes sense to impose this constraint. It should be noted that, in general, the roles played by various constituents of a symbolic structure are not necessarily identical. This is not a problem for tensor product representations, as long as the roles are known. In most cases, however, we might not even now what the roles are and how they are related. This is one of the drawbacks of the tensor product scheme which, although it allows sub-symbolic representations, does rely on a priori symbolic descriptions of knowledge.

code the symbolic structure, but are spread out on all units, as the tensor product operation adds each representation of all constituents together. Vertical weight decomposition is not, then, possible. With distributed representations, in fact, it is not even possible, in general, to recover the representations of the constituents from the representation of the structure they belong too. With the RAAM representational scheme, for example, each connectionist representation of a symbolic structure, which is obtained at the hidden layer of a non-linear network auto-associating constituents of that structure, admits an unbinding procedure that is not exact: Such an unbinding procedure, that consists in presenting the hidden pattern of activity to the second layer of the network, will not reproduce, exactly, the original constituents of the structure.

In the case of tensor product representations, the condition of linearly independency for roles does allow for the exact recovery of the representations of any constituent of a member of X, however. This has been proved by Smolensky (1987b), who has shown that recovering a filler f_i from D consists in taking the inner product of D with a vector u_i, where u_i is the unique vector of the axis orthogonal to the hyperplane spanned by all vectors of role vectors save r_i, such that $r_i.u_i = 1$. For such a vector u_i, the following equality, therefore, holds:

$$(\sum_{i=1}^{iP} f_i \otimes r_i).u_i = f_i \tag{5.1}$$

A summary of this proof is presented in appendix B at the end of this thesis.

5.3.2 Learning with fully-distributed representations

We have seen that for each network used in the previous experiments, there exists a network using fully distributed representations that would perform in exactly the same way. It does not necessarily follow, however, that such networks could learn to perform equivalently. In other words, it is not obvious at all that a network such as the one shown in figure 5.10 could, when presented with the training patterns used in the original network, learn to modify its original random weights in such a way that it could learn the required original mapping.

We investigated if such learning was possible in a number of preliminary experiments. A five layer network was first used, whose architecture was similar to the network architecture pictured in figure 5.10, with the exception that no layer of thresholding units was used. A number of experiments were performed with the domain characterized by $n = 2$, and $A = 26$, using a randomly chosen training set of $p = 50$ patterns, using the same representations for letter constituents that were used in the experiments reported in chapter 4. Since each letter constituent in a member of X does not play a privileged role within the structure it belongs to, we chose role vectors that accorded as much importance to one constituent as to the other.

In a first set of experiments, the two role vectors chosen were orthogonal:

$$\begin{aligned} r_1 &= (1,1) \\ r_2 &= (1,-1) \end{aligned}$$

The size of the hidden non-linear layer of the network used, was, as before, $H = 10$ in this case.

In a second set of experiments, the following two non orthogonal, but linearly independent, role vectors were used:

$$\begin{aligned} r_1 &= (1,2,1,0) \\ r_2 &= (0,1,2,1) \end{aligned}$$

In this case, then, the representations of the constituents of a member of the domain X overlap in a "graded" way within the representation of the member, with the representation of the first member being centered on

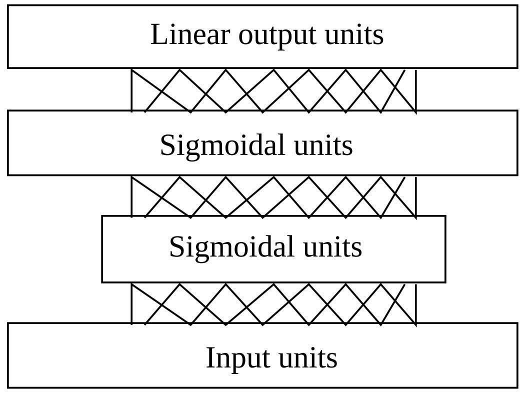


Figure 5.11: Five layer network

the first half of the input units and the representation of the second member being centered on the second half. Since such a member needed 32 units to be coded, the size of the hidden non-linear layer for the networks used was $H = 20$.

Finally, a third set of experiments were performed with the following two role vectors, similar to the two previous ones but involving less variance:

$$r_1 = (1,1,1,0)$$
$$r_2 = (0,1,1,1)$$

The size of the non-linear hidden layers used was also $H = 20$.

In all three cases, the networks used were unable to learn to criterion ($\epsilon = 0.4$) the patterns of training set. Various learning rates were used, but in all cases the gradient descent learning algorithm did not converge. The networks, in fact, could not learn the training sets when local role representations were used.

There could be a number of explanations for the learning failure of these networks. First, the topology of the energy landscape, when additional layers with different activations functions are introduced, is arguably significantly more complex, in terms of local minima, than the topology of the energy landscape corresponding to a simple three layer feed-forward network. If anything, more weights are being used.

Second, the use of a linear activation function can induce, for the weights on the connections to linear units, large weight changes, since it does not squash a very small, or very large, weighted sum of activities of a given unit. A small learning rate, therefore, needs to be used if the learning trajectory in the energy landscape is not to oscillate. Such a learning rate might not, however, be adequate to ensure large enough weights changes for weights on connectionist to non-linear units. [5]

Since using two adjacent layers of linear units does not add any more computational power than using only one layer, [6] (it might help for learning, although in most cases it probably generates more local minima in the energy landscape than anything else), a 4 layer feed-forward back-propagation was then used, consisting of a standard 3 layer feed-forward back-propagation network augmented by an additional layer of linear units at the output, as shown in figure 5.11.

The networks, this time, were able to learn the training sets, at great cost however: Over 2000 epochs were needed on average, with the same learning parameters as were used in the corresponding experiments described in chapter 4, where between 50 and 150 epochs were needed. Furthermore, the numbers of generalizations found were, on average, smaller by two orders of magnitude than the numbers found in previous corresponding experiments using local representations, and no virtual generalizations were found. Using local

[5] A possible remedy would be to use different learning rates for different layers. Or insure that the same amount of weight change is always performed, by modifying weights by a constant amount in the direction minimizing error.

[6] The two layers can be compressed into one layer computing the product of the two linear transformation implemented by each layer.

role representations, an attempt at quantitatively replicating results previously obtained also failed. In this case, the same possible explanations presented earlier in the case of five layer networks can be made.

Instead of continuing experiments with the rather specialized and arguably unrealistic and non-viable networks used in this section, we decided to directly use three layer feed-forward back-propagation networks to show that the use of fully-distributed representations would not decrease the performance observed when semi-distributed representations were used.

5.4 Auto-association with fully-distributed representations

5.4.1 Discussion

There are a number of side-effects, however, associated with the approach that consists in directly presenting a three layer feed-forward back-propagation network with fully-distributed patterns.

First, depending on the kind of representation used for the roles, the activities of patterns representing a member of X will not necessarily fall in the range $[0, 1]$, nor necessarily in the range [-1,1], when fully-distributed representations are used. In the case of sequences of two letters, for instance, the fully-distributed representation of "AB" with

$$\begin{aligned} \mathtt{A} &= \mathtt{(1,1,0,1)} \\ \mathtt{B} &= \mathtt{(0,0,1,1)} \end{aligned}$$

and the distributed representations for positional roles

$$\begin{aligned} r_1 &= \mathtt{(1,1)} \\ r_2 &= \mathtt{(1,-1)} \end{aligned}$$

will yield the fully-distributed representation

$$\mathtt{AB} = \mathtt{(1,1,1,2,1,1,-1,0)}$$

Neither the standard sigmoid activation function nor the hyperbolic tangent function

$$\mathrm{s}(x) = \frac{1}{1 + e^{-x}} \qquad \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

can therefore be used any more.

If we are to use the sigmoid, this side-effect can simply be alleviated by successively performing, for a given pattern, a translation on all activations to have them greater than or equal to 0, and a normalization to have their values in the range $[0, 1]$. A more systematic variant of this approach would be to perform the same translation, and the same normalization by a constant factor, for all patterns of The domain. The space of patterns, in this case, would thus be transformed by the following mapping

$$M : P \mapsto M(P) = \frac{P + \max_{p\in D} \max_{i=1}^{\dim(p)}(-p_i)C}{P + \max_{p\in D} \|\max_{p\in D} \max_{i=1}^{\dim(p)}(-p_i)\|_\infty C}$$

where C is the vector with all elements equal to 1, D is the space of all representations p of members of X, and $p_i, i = 1, ..., \dim(p)$ are the coordinates of p.

If this is done the transformed patterns might not, of course, be binary anymore. For the example case of the pattern "AB", for instance, a translation of 1 and a normalization will yield the representation

$$\texttt{AB} \quad = \quad (\tfrac{2}{3}, \tfrac{2}{3}, \tfrac{2}{3}, 1, \tfrac{2}{3}, \tfrac{2}{3}, 0, \tfrac{1}{3})$$

As we saw in chapter 2, we need to ensure, if all patterns are to be correctly auto-associated, that the error criterion ϵ is chosen such that for any two (distinct) patterns P_1 and P_2 of D,

$$||P_1|| - ||P_2|| \ > \ \epsilon \tag{5.2}$$

This can, unfortunately, lead to rather small values for ϵ. It should be mentioned that, for quantitative and qualitative comparisons with networks using semi-distributed representations, the constraint imposed by equation 5.2 can be ignored. As was stated in chapter 2, the networks cannot, then, be considered as proper auto-associators, but are recognizers of members of the domain, as long as satisfying discrimination from non-members of the domain is obtained. Experiments performed with such recognizers will be reported later in this chapter.

5.5 Results

In this section, we present experiments comparing performance of networks learning the domain X with n = 2, A = 20 and p = 40, when semi-distributed and fully distributed representations are used. A smaller alphabet size and training set size, relative to corresponding experiments reported in chapter 4, was chosen to allow for easier learning and better generalization. The network used was, as before, a standard 3-layer back-propagation network. Each filler representing a letter was a 16-bit random binary vector (instead of 8 in all our previous experiments), and the roles were the orthogonal vectors (1,1) and (1,-1). Each corresponding vector was translated and normalized as described in the previous section. Since all possible activation values for vectors of the domain were 0, 1/3, 2/3 and 1, the choice of $\epsilon = 0.16$ was enough to guarantee discrimination of vectors of the domain. The network, with 32 input units, 32 output units and 28 hidden units, was chosen with a small compression ratio, an increased number of hidden units facilitating learning.

Figure 5.12 compares the number of virtual generalizations and generalizations obtained with fully-distributed and semi-distributed representations. To perform the experiments corresponding to the latter, we made sure that, to ensure proper comparison, the semi-distributed representations we used had the same distribution of activities, in the same range, as the fully-distributed ones: as results shown in figure 5.13 shows (these will be discussed later), performance is quite sensitive to such a distribution.

We can see that the use of fully-distributed representations enhances performance, as a median number of 109 generalizations are obtained, versus 21 when semi-distributed representations are used. The pattern for virtual generalizations suggests that almost as many patterns that are generalized with fully-distributed representations are "only" virtual generalizations when semi-distributed representations are used. That is, they can be learned with no interference but not generalized.

Discriminations measures were performed for the networks reported above, with random patterns with activations values in the set 0, 1/3, 2/3. It is important to mention that the particular choice of the role vectors (1,1) and (1,-1) introduces a particular constraint on the form of the representations: The first half of the input units will have values between 0.0 and 2/3 while the second half will have values between 1/3 and 1. It could be that the network generalized so well on this sole constraint. We checked that this was not the case by performing our discrimination measures with random vectors obeying that constraint. It turned out that none of the 360 random vectors, in each of the five repetitions of the two experiments, was either a true or virtual generalization. A conservative lower bound for discrimination, with fully distributed representation, can be estimated by assuming that we had seen 1 true or virtual generalization in each experiment. The value of this lower bound is about $d = 4.9$.

Figure 5.13 compares the number of virtual generalizations and generalizations for different activity distributions, and shows how performance is modulated by such distributions. The first two sets of bars, from the

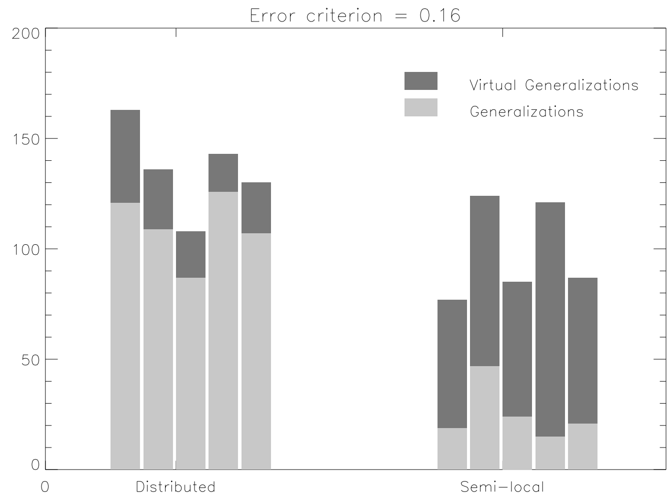


Figure 5.12: Comparison of the number of virtual generalizations and generalizations obtained with fully-distributed representations and semi-local representations, with $n = 2$, $A = 20$ and $p = 40$.

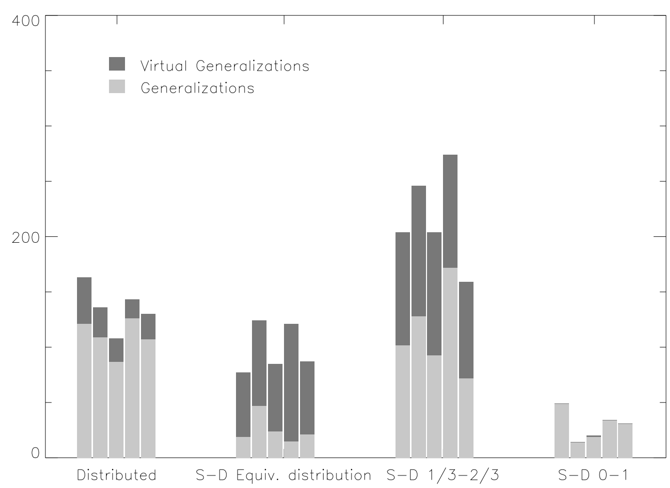


Figure 5.13: Comparison of the number of virtual generalizations and generalizations obtained with semi-local representations, with various distributions of activities.

left, indicate performance corresponding to the use of fully-distributed representations and semi-distributed representations with the same distribution of activities. The next set indicates performance for semi-distributed representations where the two role vectors were (0, 1) and (1,0) and patterns were translated and scaled so that their activities would fall in the set 1/3, 2/3. The last set of bars indicates performance when the role vectors are (0,1) and (0,1) and no translation nor scaling is performed.

We see that scaling can increase performance by almost an order of magnitude.

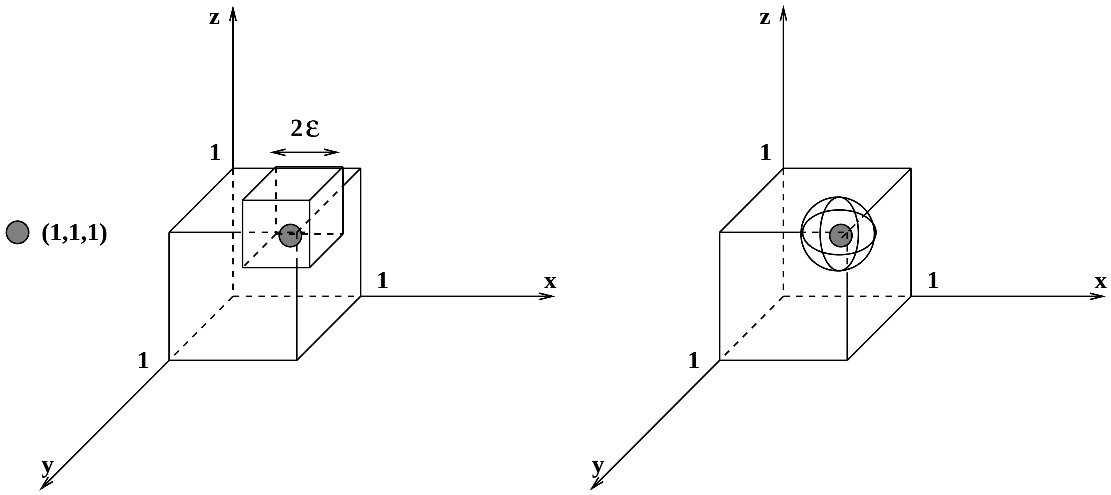


Figure 5.14: Comparison of the uses of the Euclidean norm L_2 and the norm L_∞

5.6 Recognition with fully-distributed representations

As was mentioned earlier, when the constraint on the error criterion ϵ is ignored, the networks used are not proper auto-associators, but are simply recognizers of members of the domain. To merit this label the networks, of course, need to adequately discriminate between members and non-members of the domain.

In the experiments reported in this section, we will use the standard Euclidean norm L_2, instead of the previous norm L_∞, as the use of the latter is not justified anymore and as the Euclidean norm [7] generally allows easier learning and better generalization with the back-propagation gradient descent algorithm (2.4.4).

As shown in figure 5.14, the use of the Euclidian norm now constrains an auto-associated output pattern to lie in a hypersphere of radius ϵ centered on the input pattern, instead of a hypercube centered on the input pattern and of side lengths 2ϵ.

In figure 5.15, the numbers of generalizations and virtual generalizations obtained with networks learning the domain with $n = 2$ and $A = 26$ using both semi-distributed and fully distributed representations are compared. The role vectors used were the orthogonal vectors

$$
\begin{aligned}
r_1 &= (1,1) \\
r_2 &= (1,1)
\end{aligned}
$$

in the case of fully distributed representations, and the previously used orthonormal unit vectors of R_2 in the case of semi-distributed representations. The experiments were conducted in the same conditions described in chapter 4, with $p = 50$ and $H = 10$.

The average number of generalizations is estimated at about 500. Such a number, then, yields a generalization performance of about 75%. The number of generalizations obtained with semi-distributed representations is much smaller, estimated at about 80 on average. Figure 5.16 shows the corresponding discrimination measures, which were performed in the same way as in the last experiments, with random patterns having discrete random values equal to the possible values of patterns of the training set, with the particular constraint that the first half of the input units had activation values between 0.0 and 2/3 while the second half had activation values between 1/3 and 1. We see that the discrimination of the networks using fully-distributed representations is high, with an estimated value above $d = 6$, for generalizations, compared to a value of about 1.2

[7]It should be mentioned that the argument concerning the existence of equivalent networks computing the same function with semi-distributed representations and fully-distributed representations fails when the Euclidian norm is used for error criteria, unless roles used for both experiments are orthonormal. In this case, the transformation matrices corresponding to M and M^{-1} are rotation matrices that preserve the norm of the vectors they transform, and the layer of linear threshold units (figure 5.10, which is only appropriate for the L_∞ norm, is not needed.

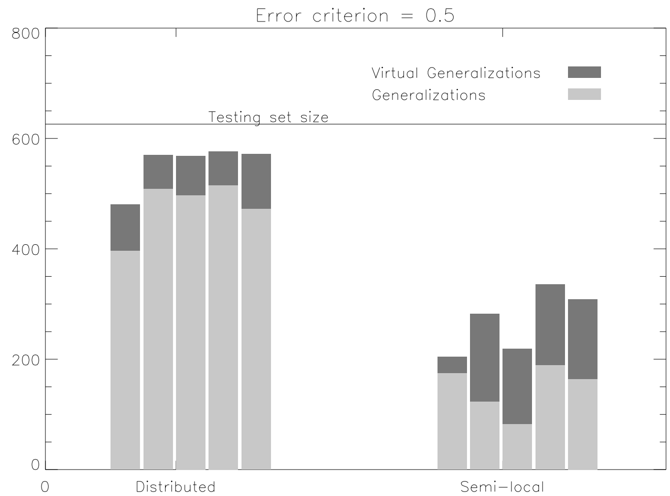


Figure 5.15: Comparison of the number of generalizations and virtual generalizations obtained with fully-distributed representations and semi-local representations, for a recognizer learning the domain with an error criterion for both training and testing was 0.5

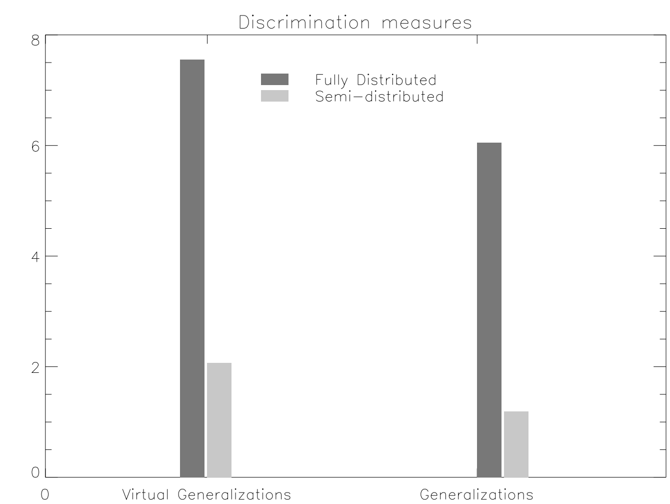


Figure 5.16: Discrimination measures virtual generalizations and generalizations obtained with fully-distributed representations and semi-local representations, for a recognizer. Error criterion for both training and testing was 1

for networks using semi-distributed representations. No virtual generalization was found in the set of 676 random patterns tested. A conservative lower bound on the discrimination power for virtual generalizations, assuming we had found one such virtual generalization in each of the five experiments, yields a value of about 7.5, compared to an estimated value of about 2 for networks using semi-distributed representations.

Since discrimination was high in these experiments, we repeated them using a looser error criterion ϵ of 1.0. We see, in figure 5.17, that the networks generalize very well, as the median number of generalizations (training patterns are excluded) has a value of 623, out of 626 possible generalizations. In two experiments, the networks generalized on 100% of the testing set. In the worse case, 614 generalizations were obtained.

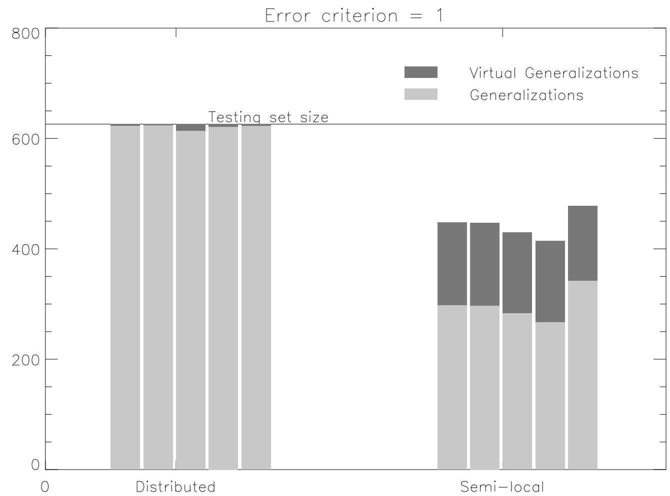


Figure 5.17: Comparison of the number of virtual generalizations and generalizations obtained with fully-distributed representations and semi-local representations, for a recognizer learning patterns of the domain with an error criterion for both training and testing of 1.0

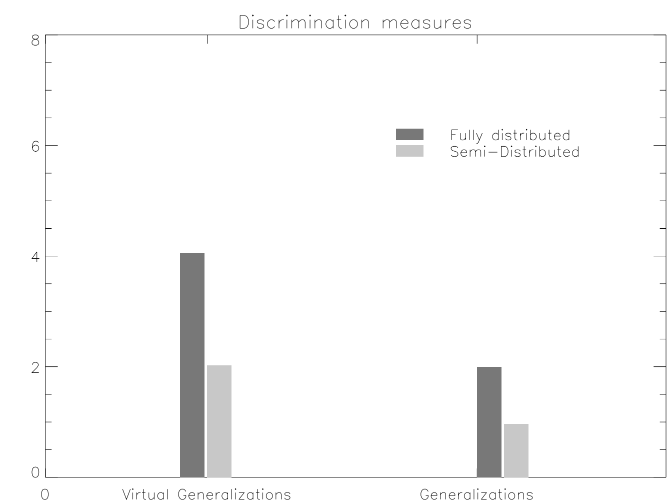


Figure 5.18: Discrimination measures for virtual generalizations and generalizations obtained with fully-distributed representations and semi-local representations, for a recognizer. The error criterion for both training and testing was 1.0

This number corresponds generalization on 98% of the testing set. For all cases, the few patterns that were not generalized were virtual generalizations.

Figure 5.18 shows corresponding discriminations measures. We see that, while it is lower than in the previous experiments, it is still just above 2 for generalizations obtained with fully distributed representations, while it is just below 1 for networks using semi-distributed representations.

Table 5.1: Some Experimental parameters for simulations performed in chapter 4

Figure	$X:n$	$X:A$	Error criterion ϵ	p
5.2	2	26	0.4	50
5.4	4	6	0.4	50
5.5	4	6 0.05	4	50
5.6	2	0.05	4	16
5.9	2	26	0.4	20
5.10	2	26	0.4	50
5.11	2	26	0.4	50
5.12	2	20	0.16	40
5.13	2	20	0.16	40
5.15	2	26	1.0 (L_2 norm)	50
5.18	2	26	0.5 (L_2 norm)	50

5.7 Conclusion

The experiments reported in this chapter suggest that the use of fully distributed representations does not decrease performance and in some cases can improve it. This suggests, then, that the use of semi-distributed representations were not crucial in the experiments reported in chapter 4, a point which was theoretically predicted in the earlier part of the chapter.

5.8 Experimental parameters

In all experiments reported in this chapter and unless explicitly stated, the learning rate used was 0.1 and the momentum was 0.9. Simulations were, as for simulations reported in chapter 3, performed with the PDP simulation package of (McClelland and Rumelhart, 1988), except for simulations involving additional layers of linear units where the simulator PlaNet (Miyata, 1990) was used. Weights were displayed using CONNEX, a graphics package developed by (Mozer, 1990). Table 5.1 summarizes relevant parameters used in the simulations.

Chapter 6

From context dependence to independence

6.1 Perspective independence

The main experiments reported in chapter 4 have shown that high systematic performance, as exemplified by massive true and virtual generalization, and generative performance, as shown by the explosive growth of true and virtual generalization when the combinatorial complexity of the domain increases, is obtained with the simple and highly combinatorial domain studied. As was mentioned in section 4.7, the numbers reported were quite small, however, compared to the potential numbers which could have been obtained had the networks **perfectly** learned the domain. And while results were dramatically improved when the alphabet size was reduced, (a ratio of over 50% generalization, and a combined ratio of true and virtual generalization of over 90% was obtained, with training set sizes making 4% of the domain with an alphabet size of 8), perfect generalization was still not obtained.

This can be explained by the fact that the networks, when learning patterns of the training set, have not been learning to independently auto-associate each letter with itself, with complete disregard to the other letters of the pattern composing its context. Using the terminology introduced by the connectionist construction of concepts theory, it can be said that they have not moved from perspective dependence to perfect perspective independence– but most certainly to a position in between. The cognitive map formation associated with learning thus occurred, as true and virtual generalization was high, but was not completed to the point where the weights learned to treat the patterns of the domain as the pure combination of sub-patterns, each corresponding to the letters composing the symbolic structures. If such a scenario had happened, the weights developed would have exhibited vertical decomposition. In that case the resulting networks would have been equivalent to the combination of individual auto-associator networks which would have each learned to auto-associate each letter of the alphabet with itself.

In this chapter, we report on experiments where the networks' weights learn such a vertical decomposition. Perfect perspective independence is thus obtained, and generalization is 100%.

6.2 Weights elimination

The notion of minimum length description was briefly mentioned in chapter 3 when measures of complexity were studied. Introduced by Rissanen (1989), such a concept is used to formalize the old idea according to which the simplest explanation or model accounting for a phenomenon is always the best ("Occam's razor"). This notion is used in the technique of weights elimination, ((Rumelhart, 1988) (Chauvin, 1990) (Weigend *et al.*, 1990) (Weigend *et al.*, 1991)) where an extra cost aimed at reducing the complexity of a back-propagation network and thus improving generalization performance, by decreasing the number of its weights, is added to the standard error cost function. The cost function on which gradient descent is made is thus, when weight elimination is performed:

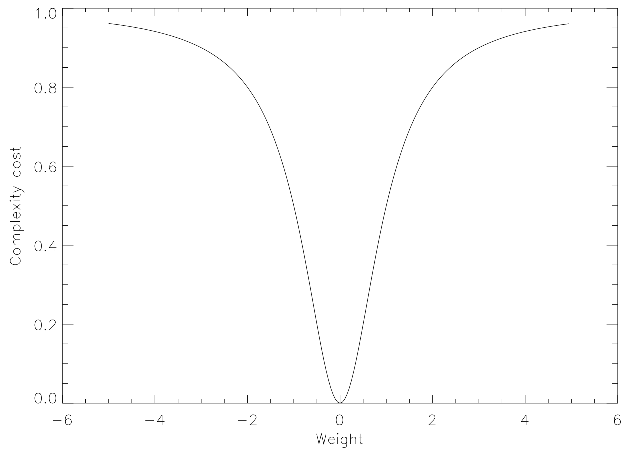


Figure 6.1: The contribution by one individual weight to the complexity cost term, when $\lambda = 1$ and $w_0 = 1$.

$$E = \sum_{patterns} E_p \; + \; \lambda \sum_{weights} \frac{\frac{w^2}{w_0^2}}{1 \; + \; \frac{w^2}{w_0^2}} \tag{6.1}$$

The first term of equation 6.1 corresponds to the standard sum of squared errors. The second term, the complexity term, is a function of all the weights of the network, and its minimization induces a reduction in the magnitude of the weights, as the contribution by a given weight to that function is close to 1 for large weights, and close to 0 for small weights. The cut-off point between large and small is controlled by the parameter w_0, which controls the width of the curve, while the parameter λ controls the ratio of the complexity cost to the total cost function to be minimized. Figure6.1 shows the complexity term , with $\lambda =$ 1 and $w_0 = 1$.

6.3 Experiment with the Cartesian product domain

A number of preliminary experiments were conducted to determine the values of the parameters λ and w_0, as well as the learning rate, momentum, and size of the hidden layer. While Weigend **et al.** (1991) use an adaptive technique to adjust λ, [1] our preliminary experiments suggested that better weights elimination was carried out with a fixed value of λ. The algorithm was also modified in that weights were thresholded during learning: Any weight whose value fell in the range [-w_{th}, $+w_{th}$], where the threshold w_{th} was determined by preliminary experiments, was set to 0 (it was still allowed to be modified after this) .

The alphabet size chosen was 15. The training sets used were of size $30n$, and it was ensured, in the random generating process, that each letter would be seen at least twice (on average, a letter was seen $2n$ times). Section 6.6 at the end of this chapter summarizes the values of experimental parameters used.

6.3.1 Performance results

Figure 6.2 shows the number of generalizations estimated, for networks learning the domain with n varying from 1 to 6, $A = 15$.

For n = 2,3, the networks were tested on the entire domain. For n = 4, 5 and 6, it was tested on 1,000 randomly generated patterns. Since all 1,000 patterns generalized correctly in the case $n = 4$, we estimate that the generalization ratio was 100%. (As the next section will show, the network implemented vertical

[1] In (Weigend *et al.*, 1991), networks are first trained with $\lambda = 0$. Once the error criterion is reached, λ is very slightly increased, unless the standard error grows, in which case λ is decreased, slightly if the long term error average is still decreasing, more dramatically if it is also growing.

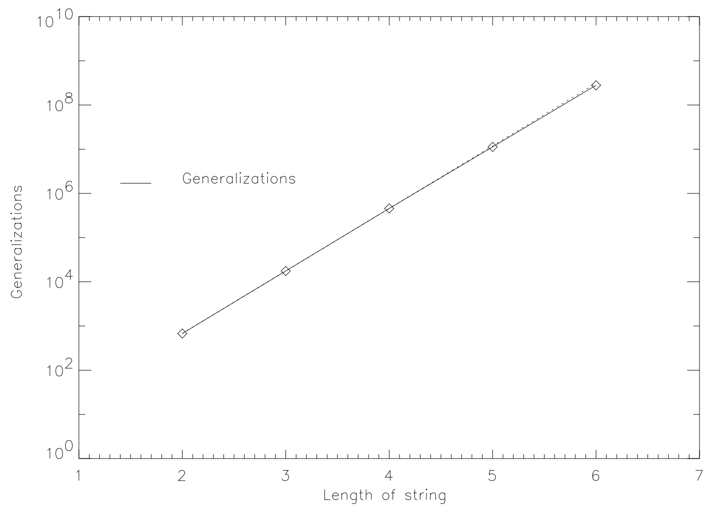


Figure 6.2: Exponential growth of generalizations for networks trained with weight elimination on sets of size 50, with $A = 26$, as n varies from 2 to 6.

decomposition. Letters were thus individually auto-associated, and a test on 1,000 patterns is enough to ensure that any pattern is correctly auto-associated). We estimate about 95% generalization for $n = 5$, and 90% for $n = 6$. The dotted line, hardly visible, is the curve $y = 15^x$, corresponding to perfect generalization.

6.3.2 Weight decomposition

Figure 6.3 shows the weights developed by the network learning the domain with $n = 2$. Vertical decomposition is observed, as each row of the lower matrix, corresponding to a hidden unit, contains non-zero weights on either the first 8 elements corresponding to the first group of 8 units coding the first letter of the pattern, or the second 8 consecutive elements corresponding to the second 8 units coding the second letter of the pattern. The same property is observed on the columns of the upper matrix.

Figure 6.4 shows the weights developed by the network learning the domain with $n = 4$. r

Weight decomposition, again, can be observed. Since $n = 4$, a given hidden unit will respond to one in four groups of 8 input input units (or output units) coding an individual letter in a string of the domain. Figure 6.5 shows the weights developed by the network learning the domain with $n = 6$.

Vertical decomposition, again, is obtained, although it is not perfect, as a few hidden units map to more that one of the six pools of input or output units coding a given letter in the strings of the domain.

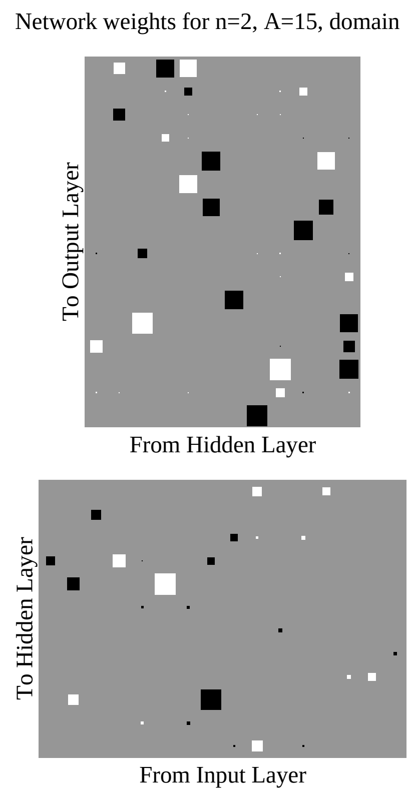


Figure 6.3: Systematicity through vertical weight decomposition, for a network with n =2 and A = 15. Only the weights of the network connected to units involved in the coding of several letters at the same position are non-zero. Context independency is thus achieved.

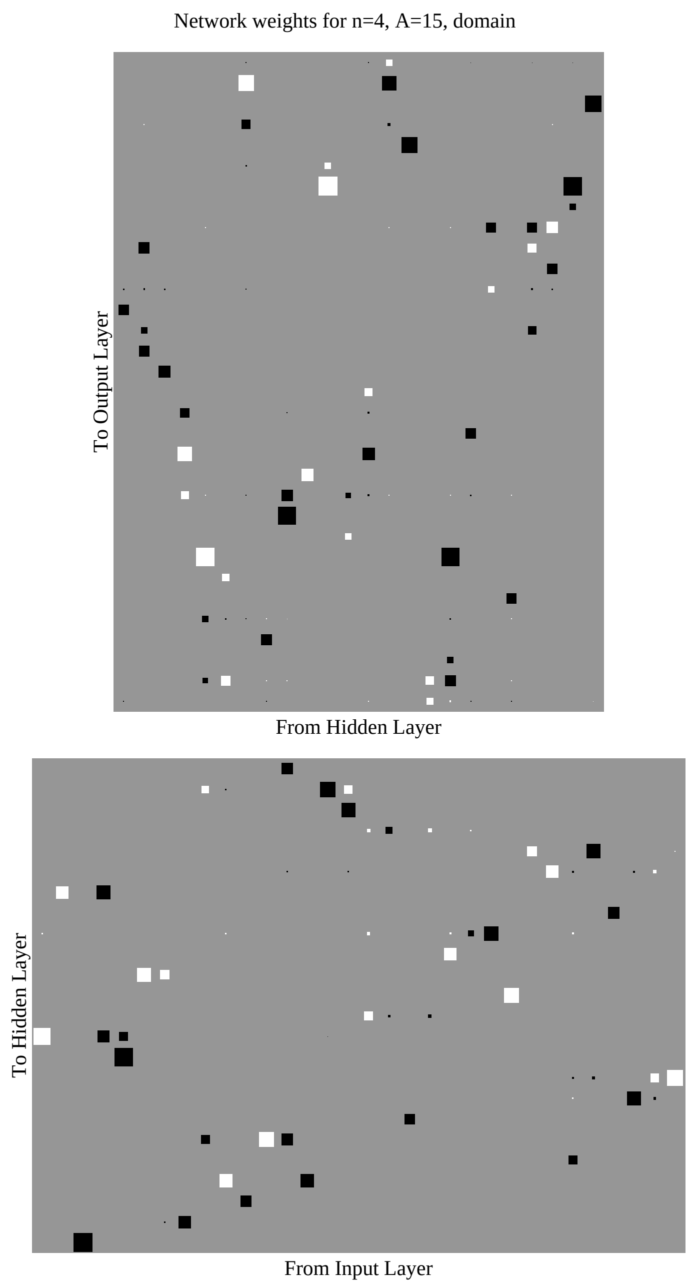


Figure 6.4: Systematicity through vertical weights decomposition, for a network with n =4 and A = 15.

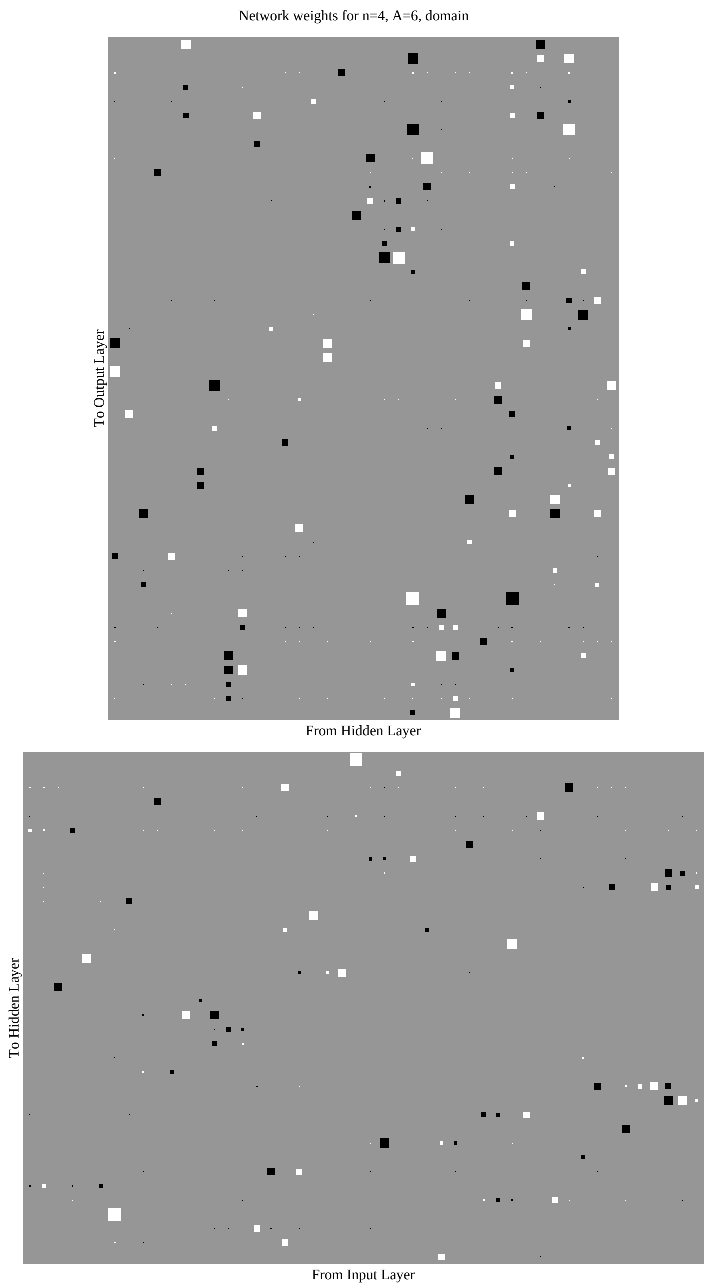


Figure 6.5: Systematicity through vertical weights decomposition, for a network with n =6 and A = 15. The decomposition is close to perfect.

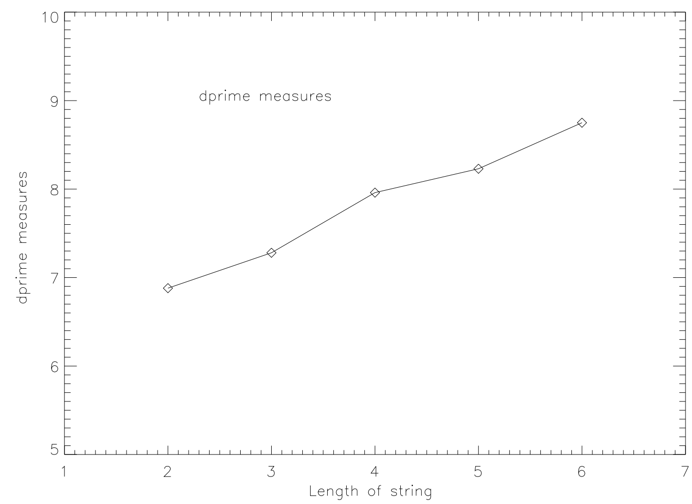


Figure 6.6: Discrimination measures d' for the networks of figure 6.2.

6.3.3 Discrimination results

To measure how the networks learned to discriminate between members and non members of the domain, the d' measure was used, instead of the previously used d measure which was infinitely high when the network generalized on all members of the domain.

Figure 6.6 shows the values of d' for the various experiments, which are significantly high.

This measures correspond to the discrimination ratios listed in table 6.1.

6.4 Experiments with English words

In this section we report on the harder problem of extracting the semi-combinatorial structure of sets of written English words. The domains studied consisted of the sets of English words of length n, $n = 3, \ldots, 6$ [2]. Networks were trained on an arbitrarily chosen subset of 100 words, with weight elimination, for 10,000 epochs. The testing criterion ϵ was chosen to be small (0.2) as higher discrimination was obtained this way.

6.4.1 Performance results

Figure 6.7 shows the number of generalizations obtained, as n varies from 3 to 6. Performance is high, as more than 90% of the testing set generalizes if n is greater than 3.

6.4.2 Discrimination results

Figure 6.8 shows d values measuring how the networks have learned to discriminate between the set of English n-letter words and the set of n-letter strings.

Discrimination is surprisingly high, as values greater than 5 are obtained for networks learning the domains with n greater than 3.

6.5 Concluding remarks

The experiments reported in section 6.3 involving the Cartesian product of sets, are a very simple yet good example, we believe, of what the formation of a cognitive map, implemented via a move from perspective

[2] The words used were those found in the "spell" dictionary available on Unix systems.

Table 6.1: Ratios of probability of generalizations for members of X and random binary patterns

n	2	3	4	5	6
Discrimination ratio	5.3	9.8	27.8	83.3	333

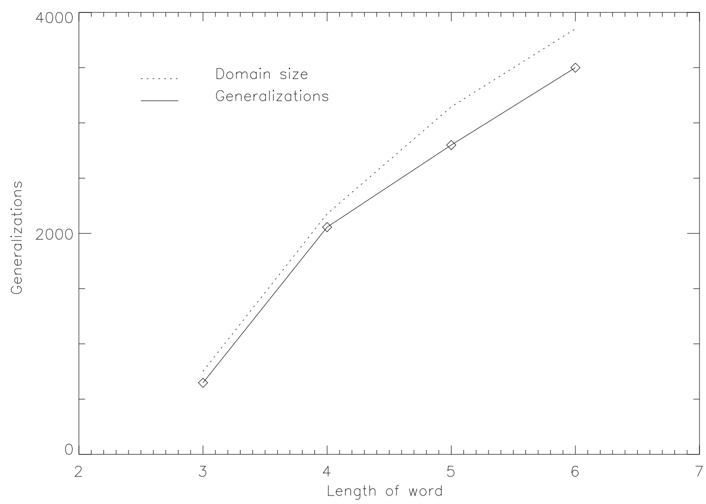


Figure 6.7: The number of generalizations obtained for networks trained on subsets size 100 of English n-letter words, as n varies from 3 to 6.

dependence to perspective independence, can amount to. The statistical regularity of the training sets allows the networks to rely on their combinatorics, and thus view them and process them in a systematic way.

Experiments performed with English words show high performance, too, and unusually high discrimination. It could be, however, that the networks did not rely on the orthographical regularity of the sets but simply learned vertical weights decomposition on any letter present in the strings of the training set. High discrimination measures, in this case, could be explained by the simple fact that some letters were never seen in the training sample, (the letter “z” in the second position of a 3 letter word, for instance.).

6.6 Experiments parameters

λ was 0.0008 for networks learning the domain with n = 2,3 and 4, and 0.0004 for networks learning the domain with $n = 5$ and 6. The sizes of the hidden layers were slightly increased from the ones used in the main experiments reported in chapter 4, to $H = 6n$, to ease the independent auto-association of individual letters. The learning rate was 0.1, except for the cases corresponding to n = 5, 6 where it was 0.05. The momentum was 0.1. Weights were thresholded at 0.01. Training was stopped after 10,000 epochs. The error criterion for testing was $\epsilon = 0.4$. Input and target patterns had activities of 0.1 and 0.9, the representations used being the same semi-distributed representations which were described in chapter 2. The same parameters were used for experiments involving English words. Weight thresholding was not, however, performed in that case. The testing set criterion ϵ_t was 0.2.

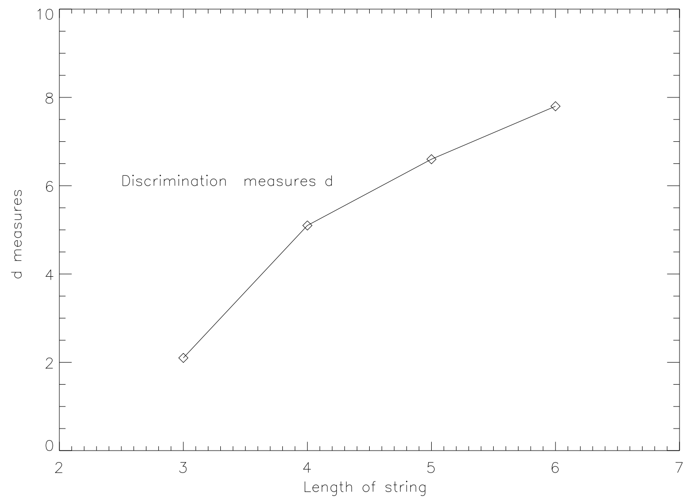


Figure 6.8: Discrimination measures d for the networks learning the domain consisting of English n-letter words, as n varies from 3 to 6. The set to be discriminated from is the set of all n-letter strings.

Chapter 7

Analysis and interpretation

7.1 Overview

In this chapter, a formal analysis in the case of one-layer feed-forward back-propagation networks learning to auto-associate representations of elements of the combinatorial domain

$$X^N = \{(x_1,\ x_2\ , ...,\ x_n) \mid x_i\ \in\ X\}$$

introduced earlier is attempted. Such an analysis provides a first step towards understanding how the networks involved in the experiments reported in chapter 4 develop their weights and their associated competence. Statistical analyses, in the more general and complex 2-layer case are then performed.

7.2 Analysis in the one-layer case

7.2.1 Formal analysis

Linear networks [1] learning the combinatorial domain X, with $||X|| = A$, where elements are represented with purely local representations involving orthogonal role representations are first considered.

Notation
Without loss of generality, the filler vectors $(f_1, f_2, ..., f_A)$ representing each element of X can be chosen as

[1] There is a simple property induced by tensor products in the linear case:

$$\forall\ a_1a_2..x..y..a_n \in X$$

$$\forall\ a_1a_2..x..a_n \in X,\ \ \forall\ a_1a_2..y..a_n \in X,\quad \forall\ a_1a_2..a_n \in X$$

then

$$a_1a_2..x..y..a_n = a_1a_2..x..a_n + a_1a_2..y..a_n - a_1a_2..a_n$$

For sequences of length 2, for instance, the vector representing ad can be rewritten as the sum of vectors ab + cd - cb. This property was investigated, but proved to be inconclusive. The related study is presented in appendix C.

unit vectors of the canonical orthonormal basis of R^A.
Likewise, the role vectors $(r_1, r_2, ..., r_n)$ representing the position i of an element of X in an n-tuple of X^n can be chosen as unit vectors of the canonical orthonormal basis of R^n.
Let the training set consist of P elements $(x_{1p},\ x_{2p},\ ...,\ x_{np})$, $1 \le p \le P$.
These elements are represented by the $n \times A$ dimensional tensor product vectors

$$x_p = \sum_{i=1}^{n} f_{ip} \otimes r_i$$

If $(v_1, v_2, ..., v_{A\times n})$ is the canonical orthonormal basis of R^{nA}, then

$$x_p = \sum_{i=1}^{n} v_{ip+A(i-1)}$$

Let $\mathcal{P} = \{1, \ldots, P\}$ and $\mathcal{V} = \{1, \ldots, nA\}$, and let $W \in R^{nA} \times R^{nA}$ be the matrix of weights $w_{j,k}$ of a network having learned, with the delta rule, the P patterns of the training set. (We will show that such a matrix exists.) Initial weights are assumed, for now, to be null.
The total error E in term of least squares is defined by:

$$E = \sum_{p=0}^{P} (x_p - Wx_p)^2$$

The following equations hold:

$$E = 0$$

$$\Leftrightarrow \forall p \in \mathcal{P}, Wx_p = x_p$$

$$\Leftrightarrow \forall p \in \mathcal{P}, W\sum_{i=1}^{n} v_{ip+A(i-1)} = \sum_{i=1}^{n} v_{ip+A(i-1)}$$

$$\Rightarrow \forall (j,p) \in \mathcal{V} \times \mathcal{P} \begin{cases} \sum_{i=1}^{n} w_{j,ip+A(i-1)} = 1 & \text{if `}\exists i|\ \ j = ip + A(i-1) \\ \sum_{i=1}^{n} w_{j,ip+A(i-1)} = 0 & \text{otherwise} \end{cases} \tag{7.1}$$

Equation 7.1 indicates that, for a given row of the weight matrix, only those weights that are on connections leading to units that can be active need be non zero. Since the linear delta rule updates a weight matrix W_{old} according to

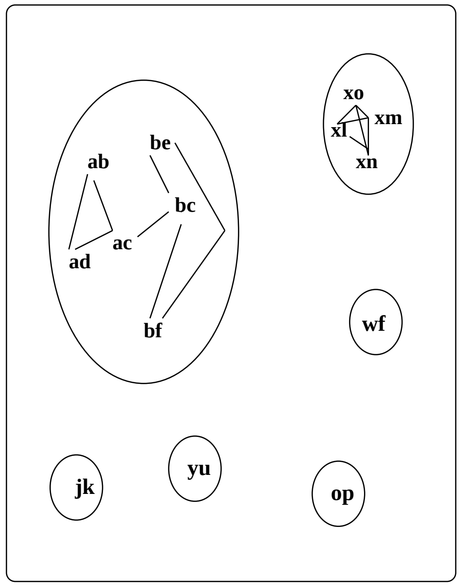


Figure 7.1: The partition induced by the relation R for the set of training patterns {ab, ad, ac, bc, be, bf, xo, xl, xn, xm, jk, yu, op, wf}.

$$W_{new} \;=\; W_{old} \;-\; \eta\,(W_{old}x_p - x_p)x_p^T$$

for a given pattern x_p (assuming $W_{initial} = 0$, any weight which is on a connection to a unit that is not turned on by any pattern of the training set will remain null.

To solve equation 7.1, it is helpful to distinguish between the different kinds of relations that patterns of the training set might have with each other. The following relation of equivalence among patterns, R, is introduced: Let the letter patterns corresponding to the representation vectors X_p be vertices in a graph, connecting two vertices if an only if one has a letter in common with the other. Let R be the binary relation defined by, if S_1 and S_2 are two members of the training set in X^n:

$$S_1 \; R \; S_2 \Leftrightarrow S_1 \text{ is connected to } S_2 \text{ via a path in the graph}$$

R is a relation of equivalence, and its equivalence classes partition the training set into connected sub-graphs. Figure 7.1 shows the graph just defined, and the equivalence classes for a training set of patterns for two letters: {ab, ad, ac, bc, be, bf, xo, xl, xn, xm, jk, yu, op, wf}. For clarity, all arcs between a pattern and itself are not represented. All patterns in an equivalence class will contribute a number of equations in 7.1 independent of any equations contributed by patterns in another equivalence class, since they will not share any common letter (common connection) with patterns of another class. It is thus possible to study separately the solution for each equivalence class. Three kinds of equivalence classes can be distinguished:

(1) The singleton equivalence classes.
In this case, the single pattern in the equivalence class has no letter in common, or shares no common "on" units, with any other pattern of the training set. The sub-matrix of weight solutions to the subset of equations making 7.1, defined by those weights, and those only, which connect units coding the letters of the pattern, is a square matrix of n^2 weights whose sum, on each row, needs to be 1. An obvious and most symmetrical solution for the value of each of these weights is $1/n$. Figure 7.2 shows that the weights do find this solution, obtained in a simulation.

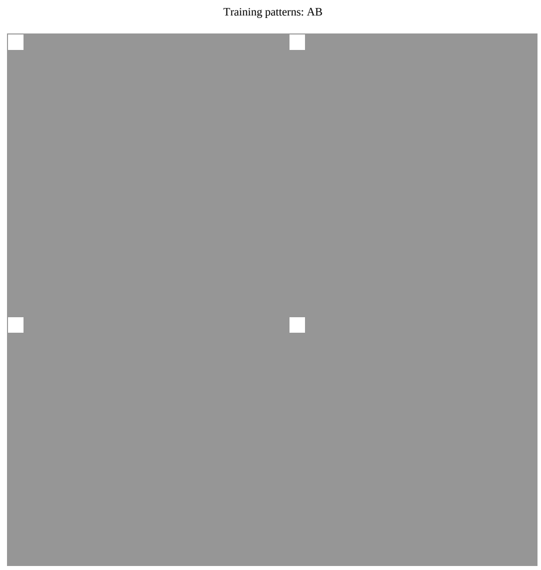


Figure 7.2: The weights developed by a linear network learning the individual pattern ab.

(2) Equivalent classes where all patterns are connected to all others.

This is the case of the equivalence class formed by {xo, xl, xn, xm } in figure 7.1. There is a simple, most symmetrical solution to this case, which, in the case of one common letter and q patterns, is the following matrix:

$$\frac{1}{q+1}\left(\begin{array}{cccc} \mathrm{q} & 1 & 1 & 1 \\ & & & \\ & & & \\ \mathrm{q} & 1 & -q & -q \\ & -q & 1 & -q \\ & -q & -q & 1 \end{array}\right)$$

where the "weights" on the left side of the matrix correspond to the common letter, and the weights of the right side correspond to the other letters. Such a solution is also found experimentally and depicted by the weight matrix of figure 7.3, which shows weights developed by a linear network having learned the patterns {ab, ac, ad}.

(3) Equivalent classes where not all patterns are connected to all others.

This case can in fact be considered as the superposition of the two cases earlier discussed, (except for the magnitude of the weights) where patterns have common letters. Figure 7.4 shows the weights of a network having learned the patterns {ab,ac,ad,bc,be,bf}. The weights are a superposition, within a factor, of the earlier case depicted in figure 7.3 and the case where weights would have been developed with the training patterns bc, be, bf, as shown in figure 7.5. It is interesting to note that the weights find the most simple and symmetrical solution, in spite of the asymmetry of the training set introduced by the pattern bc.

Figure 7.6 shows the weights developed by a network having learned 40 patterns, with $n = 2$ and $A = 16$. The clear division in 4 different regions corresponds to the superposition of the three cases distinguished earlier. The error criterion was $\epsilon = 0.4$. Figure 7.7 shows the weights developed by a network having learned 40 patterns, with $n = 4$ and $A = 18$. The weights of a network having learned in the limit, that is, having

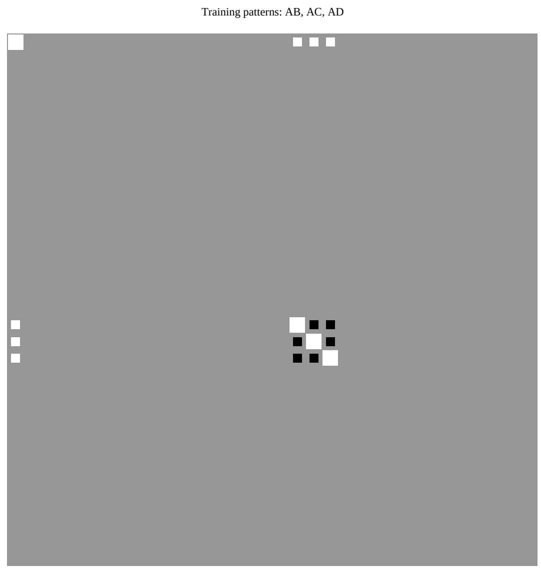


Figure 7.3: Weights developed when learning the patterns ab, ac, and ad.

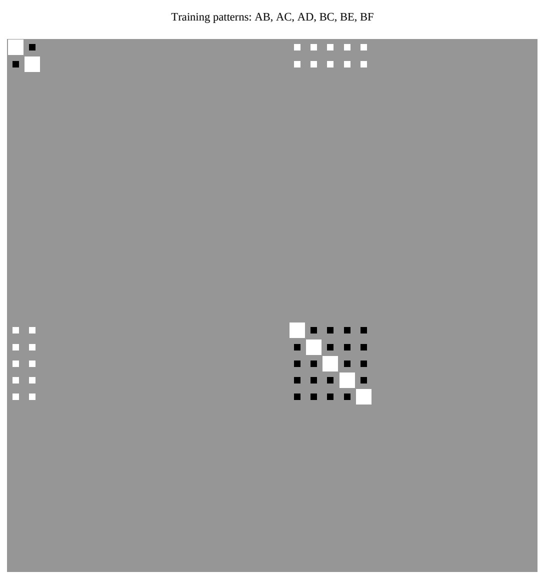


Figure 7.4: Weights developed by a network learning the patterns ab, ac, ad, bc, be, bf.

learned all patterns of the domain with a harsh error criterion of 0.0001, are shown in figure 7.8.

The non-linear case

The use of a non-linear activation function, as shown in figure 7.9 which compared weight matrices obtained with different activation functions, does not change our explanation significantly, a sigmoidal function simply pushing weights to larger values.

The weight matrix shown on the upper left corner of the figure is the same as the one shown in figure 7.6, and corresponds to a network which has partially learned with a linear activation function the combinatorial

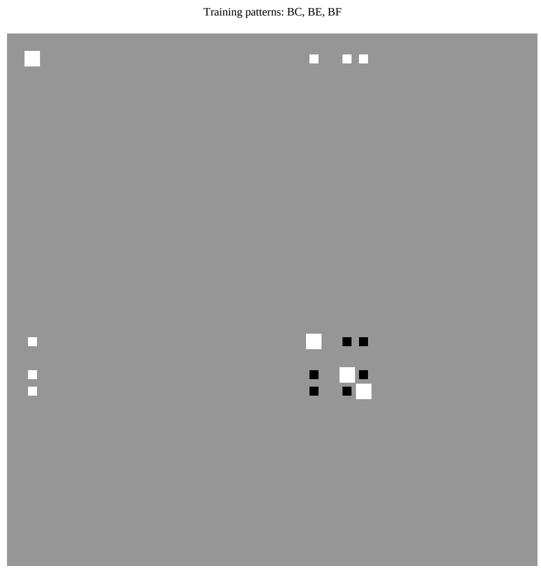


Figure 7.5: Weights developed by a network learning the patterns bc, be and bf.

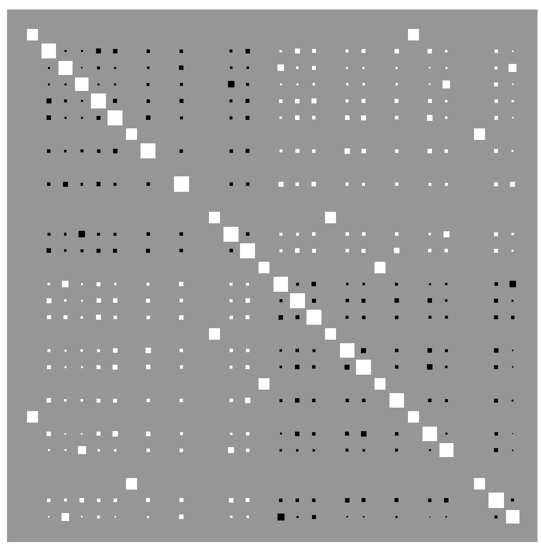


Figure 7.6: Weights developed by a network having learned 40 patterns, with $n = 2$ and $A = 16$.

domain with $n = 2$, $A = 16$ and $p = 40$.

The weight matrix shown on the upper right corner of the figure corresponds to weights obtained with the the standard logistic activation function, for the same problem. We observe strong similarities, with large positive weight values (white squares) being present in both matrices.

The weight matrix shown on the lower left corner of the figure corresponds to weights learned with the tanh activation function. Proper scaling of these weights, yielding the weight matrix shown on the lower right corner of the figure, again reveals similarities with the two previous weight matrices.

Non-local representations

When non-local representations are used, as was the case in all the experiments reported in this thesis, a

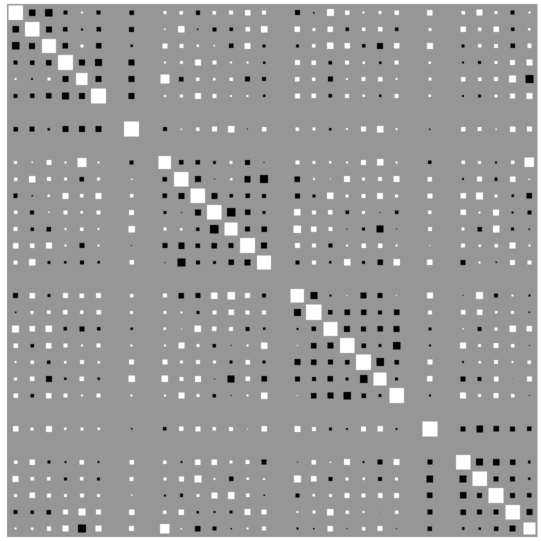


Figure 7.7: Weights developed by a network having learned 40 patterns, with $n = 4$ and $A = 8$.

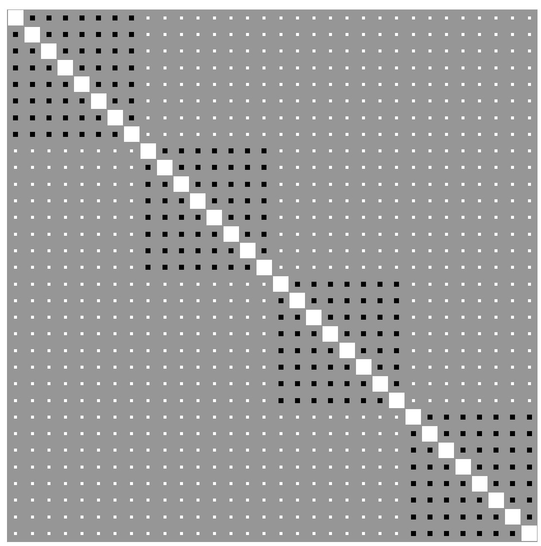


Figure 7.8: The weights of a network having learned all patterns of the domain with a harsh error criterion of 0.0001.

given unit might participate in the coding of several letters. The same reasoning that has been carried out can now be made if one considers that the network is learning binary patterns, where each unit codes a one and only one letter in any position. The submatrices corresponding to the various equivalence classes now have elements which are not contiguous withing the larger matrix. However, because of the mere statistical regularity in patterns of the training set, we can expect the network to develop weights exhibiting the same kind of spatial partitioning as those shown in the last figure, for instance, if and only if the number of patterns in the training set is large.

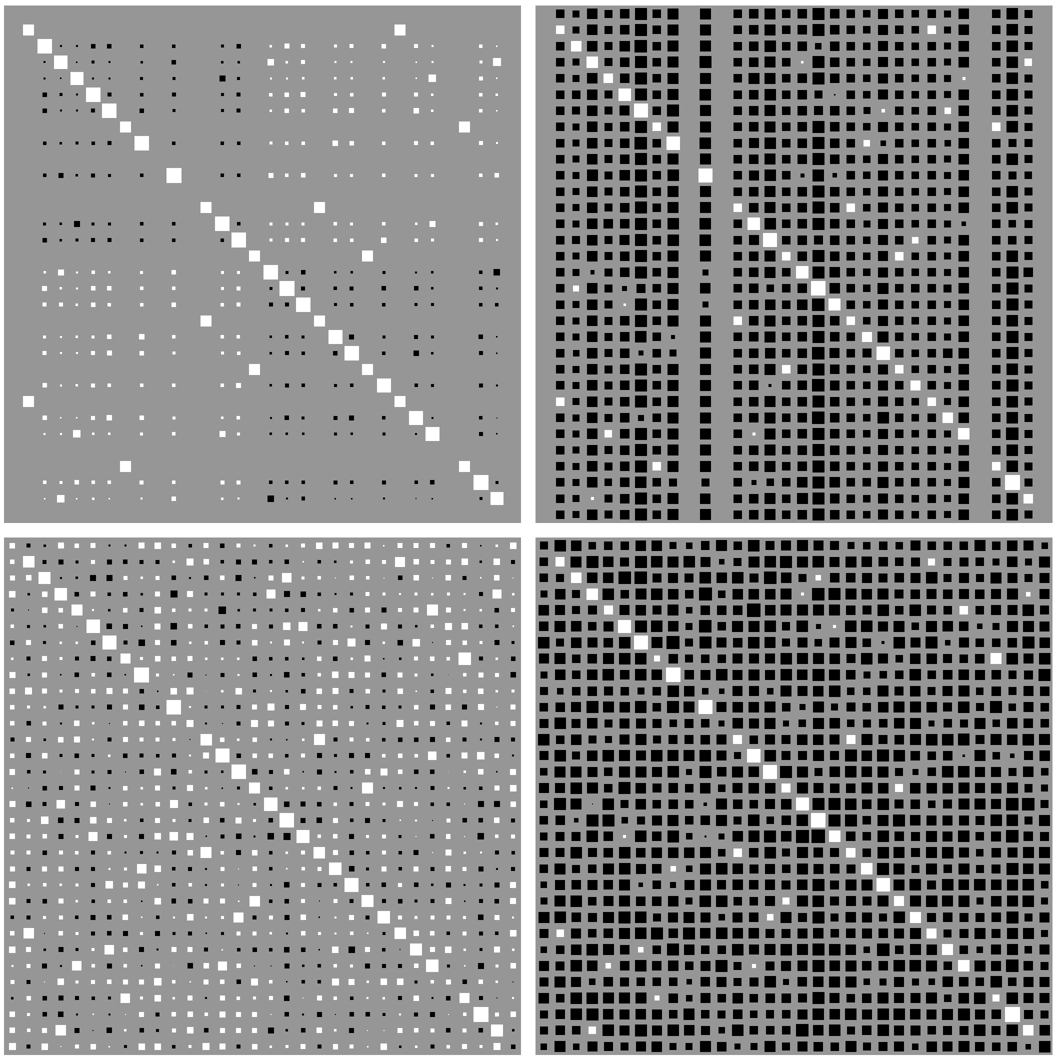


Figure 7.9: The weights of networks having partially learned the domain with $n = 2$, $A = 16$, and $p = 40$, for different activation functions. The upper left weights correspond to a linear activation function, the upper right to the standard logistic, and the lower left to the tanh function. The lower right weights are scaled from the lower left weights, to show the similarity with for all three activation function used.

7.3 Statistical analyses

7.3.1 Introduction

The first part of this chapter gave us some information on how weights could develop in one layer-networks learning the combinatorial domain we have been concerned with, but fell short of making significant predictions on how the multi-layer networks reported on in the previous chapters developed their weights. While we have seen that the technique of weight elimination allows the networks to implement a most simple and visible vertical decomposition solving the combinatorial problem, networks trained with the standard delta rule exhibit weights that do not present any apparent structure.

The rest of this chapter is devoted to testing the hypothesis according to which the networks, shy of exhibiting perfect systematicity and generativity but nonetheless displaying massive true and virtual generalization, fell somewhere in between the following two extremes: One corresponding to pure rote learning of the training set (complete perspective dependence and absence of a cognitive map) and one corresponding to perfect vertical weight decomposition (complete perspective independence and formation of a cognitive map). If this hypothesis is correct, then the hidden units should, in a systematic manner, respond to the presence or absence of patterns[2] of subpatterns corresponding to constituents of the strings, rather than respond to arbitrary subpatterns in the representations of the strings.

While mathematical analyses of multi-layer feed-forward networks have started to appear (Baldi and Hornik, 1988) (Goggin *et al.*, Submitted), the simpler approach using statistical techniques was used. Three of these were used to test our hypothesis: Cluster analysis, principal component analysis, and contribution analysis.

Cluster analysis

Cluster analysis (e.g (Everitt, 1980)) analyzes sets of vectors and clusters them according to their similarity or closeness in space. This is done by generating a symmetrical matrix of similarity between vectors, where the element in row i and column j is the distance between the ith vector and the jth vector, and iteratively merging the two most similar vectors into a cluster. [3] Euclidian distances are usually used as a similarity measure, in which case the similarity matrix is the matrix of distances between any two vectors. The clusters are usually represented in a dendogram, where vertical height corresponds to the distance between clusters being merged.

This technique has proved to be useful in analyzing hidden units activities in connectionist models (Rosenberg, 1987) (Elman, 1990) (Servan-Schreiber *et al.*, 1989), although it does have one drawback: it does not reveal **how** vectors are similar. Two vectors identical except in one of their element, for instance, might be far apart if the difference between their corresponding element is very large.

Principal component analysis

Principal component analysis (e.g (McDonald, 1985)) is a statistical technique which attempts to account for the greatest variance of a set of vectors. New axes, the principal components, are found such that the greatest variance in the set of vectors lies in the direction of the first principal component, the next greatest variance lies in the direction of the second principal component, and so on until the smallest variance lies in the direction of the last component. The principal components are the eigenvectors of the covariance matrix, and the amount of variance factored in by the eigenvectors are the corresponding eigenvalues. If the matrix formed by the set of vectors is multiplied by the matrix of eigenvectors (ordered by descending eigenvalues), the first column of the resulting rotated matrix, which will be referred to as the first rotated variable, displays the highest separation possible among the points. The second column displays less separation, and so on. Principal component analysis is thus a method to account for the variance of a large number of variables

[2] The word here is used with its traditional meaning, as in pattern recognition.

[3] The smallest element of the matrix is found (the diagonal is ignored), and the two vectors corresponding to that element are clustered. The two rows and columns in the similarity matrix corresponding to the vectors are merged into one (the largest value is selected for each dimension), these rows and columns are added in the matrix, while the two rows and columns corresponding to the two previous vectors are deleted from the matrix. The clustering procedure is iteratively repeated until there is only one cluster.

Table 7.1: True and virtual generalizations, and discrimination measure, for the network analyzed.

Size of the domain	1,296
Size of the training set	50
Generalizations	316 (24.3%)
Virtual generalizations	642 (49.5%)
d measure: generalizations	6.35
d measure: virtual generalization	6.46

with a smaller number of factors, that number being determined by how many components one takes into consideration.

Like cluster analysis, this technique has been used in a number of connectionist studies, mostly to analyze vectors of hidden unit activities (Rosenberg, 1987) (Elman, 1990)).

Contribution analysis

Unlike the two statistical techniques just described, contribution analysis ((Sanger, 1989) (Sanger, 1990)) is specific to neural networks. For a given input pattern presentation, hidden unit, and output unit, the contribution of the hidden unit is defined as the product of the activity of the hidden unit and the weight from the hidden unit to the output unit. This definition is slightly modified in that the sign of the contribution is adjusted to reflect the correctness of the contribution towards pushing the output unit in the right direction. Contributions in the right direction are thus positive, while contributions in the wrong direction are negative.

Contribution analysis consists in performing principal component analysis on various two-dimensional slices of the three-dimensional matrix of contribution. A more complete discussion of this method is presented in section 7.7.

7.3.2 Networks used

The network on which cluster, principal component, and contribution analysis was performed [4] was a network having learned the combinatorial domain X with $A = 6$ and $N = 4$. The small alphabet size allowed for significant true and virtual generalization, and a large enough domain consisting of 1,296 elements. The network was trained on an arbitrarily chosen subset of 50 patterns. The training error criterion was $\epsilon = 0.4$ as in experiments reported in chapter 4. Table 7.1 displays the relevant information concerning the performance of the network. Figure 7.10 shows the weights developed by the network, which do not show any obvious structure.

[4] It should be mentioned that that none of the analytical methods used is specific to the problem studied, involving combinatorial domains.

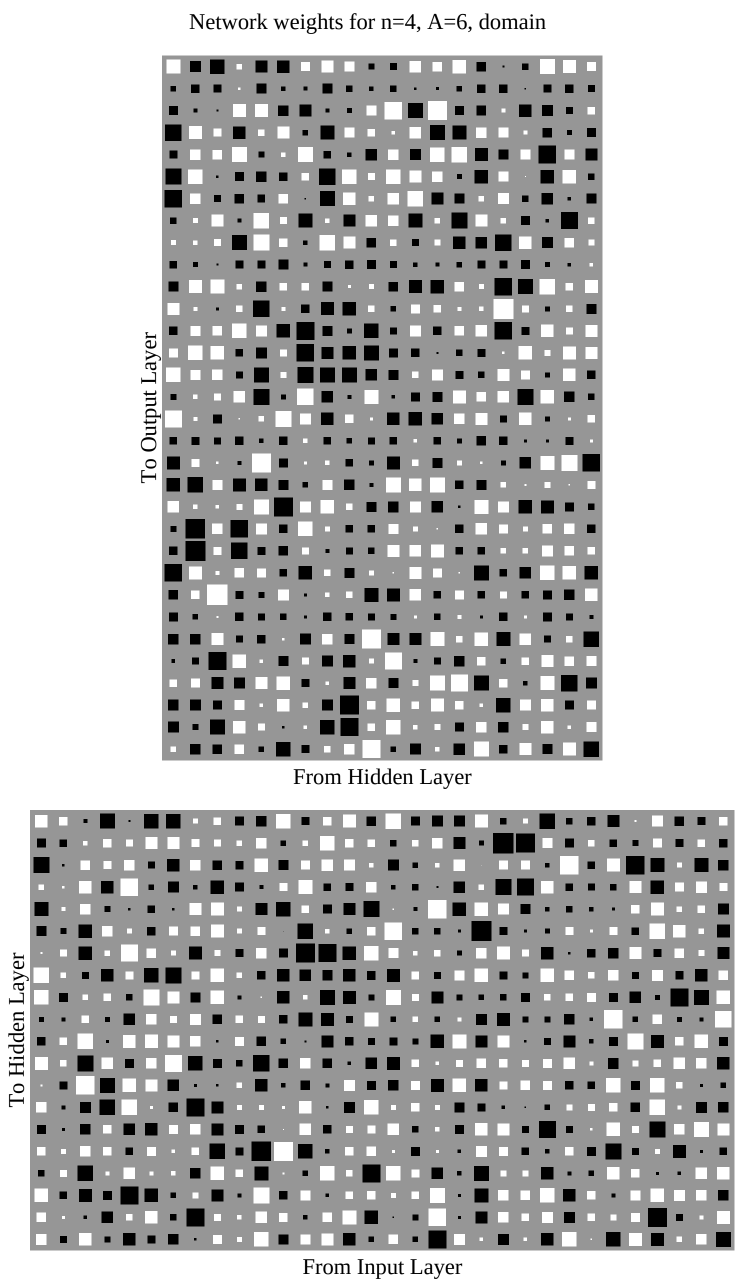


Figure 7.10: The weights of the network used analyzed. No structure is apparent.

7.4 Analysis of the weights

In this section, we report on analyses performed on the weight matrices of the network studied. While our expectations of observing meaningful vector clusterings was low (there is no reason to believe that such analyses could reveal anything, even if they were performed in the "best" case, that is in the case of vertically decomposed weights), they are partially reported here for comparison purposes.

7.4.1 Cluster analysis

Figure 7.11 shows the clustering tree corresponding to the vectors making the rows of the first layer weights matrix. No pattern seems visible here, expect for least informative fact that the weights seem to be quite "unclustered".

Figure 7.12 shows the clustering tree corresponding to the vectors making the columns of the matrix. Again, there are very few comments that can be made about the tree. The same analysis was performed on the matrix of weights of the second layer of the network. Such an analysis, again, gave rise to no interpretation.

7.5 Principal component analysis

Principal component analysis was also performed on both the rows and columns of the two weights matrices of the network studied, but did not lead to any intesting interpretation.

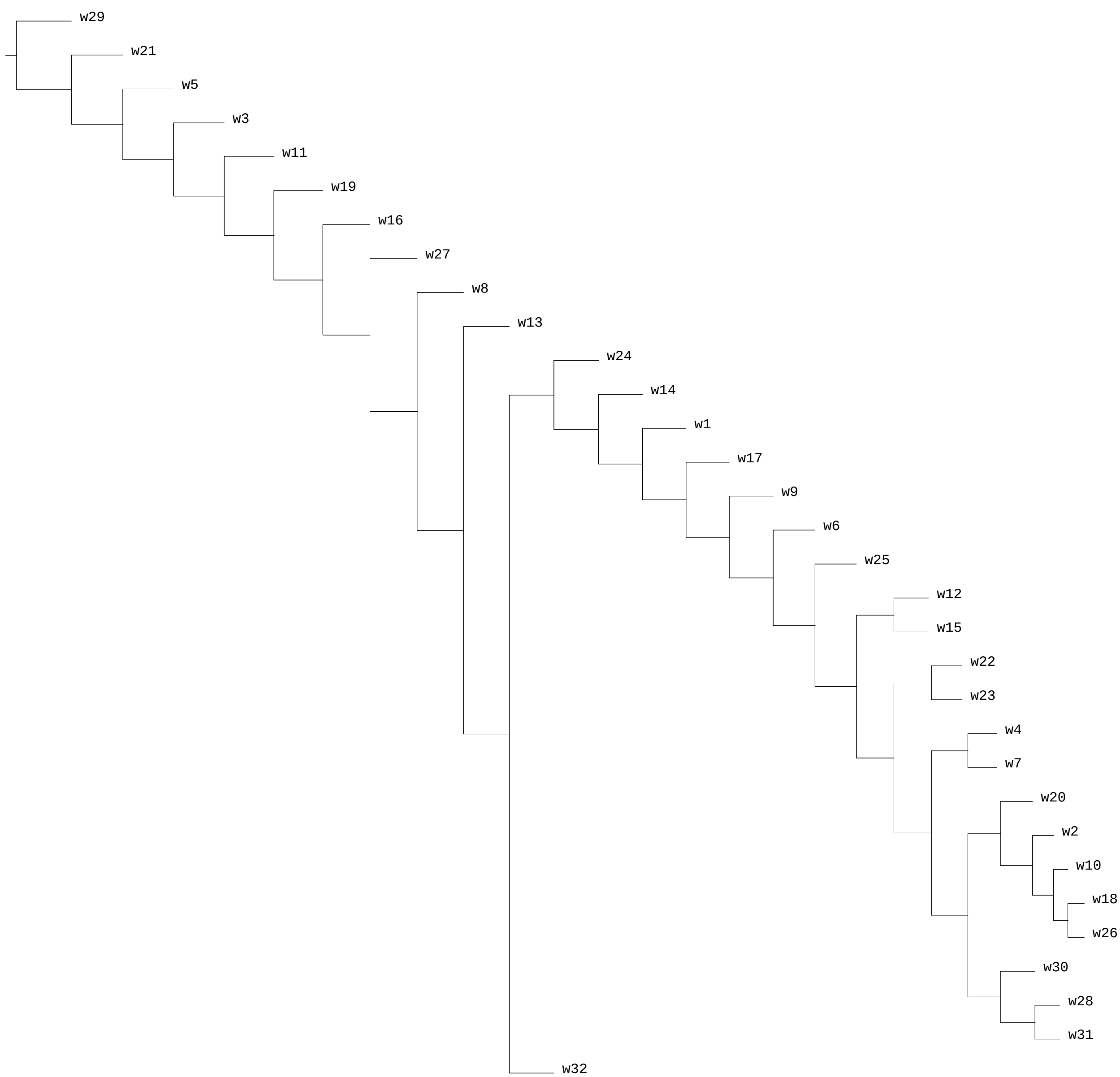


Figure 7.11: Clustering tree for first layer weights. No conclusions can be made.

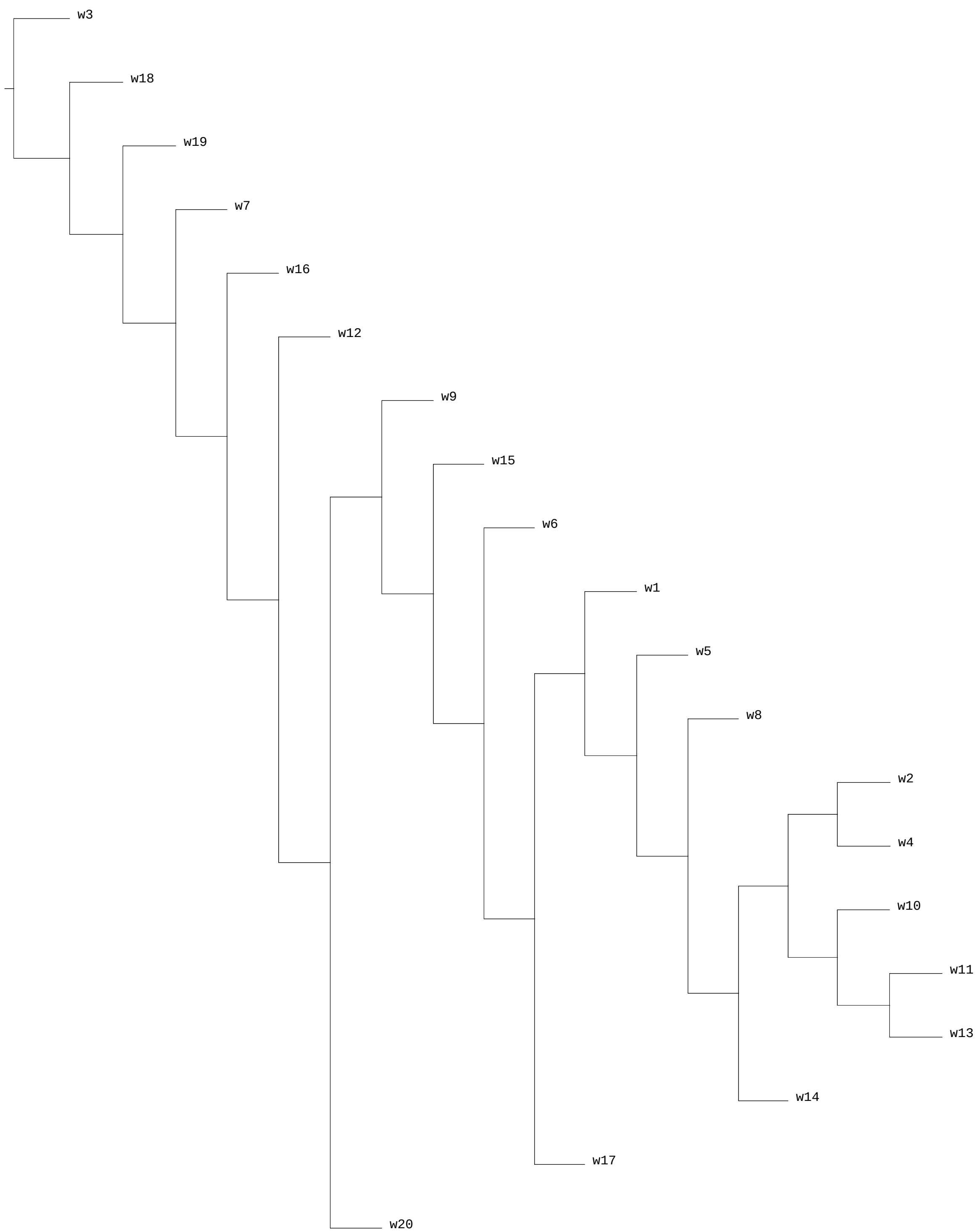


Figure 7.12: Clustering tree for first layer weights (transpose).

7.6 Analysis of the hidden unit activities

Cluster and principal component analysis are used, in this section, to analyze vectors of hidden unit activities.

7.6.1 Cluster analysis

Figure 7.13 shows the clustering tree corresponding to the vectors of hidden unit activities when each pattern of the training set is presented to the network. The nodes of the tree are labeled by the corresponding pattern. We observe that the patterns clustering together tend to contain the same sub-strings. The cluster of depth 3 on the lower right corner of the figure, for instance, clusters strings that all have the letter "f" in the first position, and the subcluster of depth 2 clusters strings that have the subpatter "ff" in the last two positions.

Figure 7.14 shows the clustering tree corresponding to hidden units activities average over all patterns of the domain, for a given letter at a given position; that is, all vectors of hidden unit activities corresponding to input patterns of the domain having the same letter in the same position were averaged together. This averaging method, commonly used when connectionist models are analyzed, does not, however, yield any interesting result.

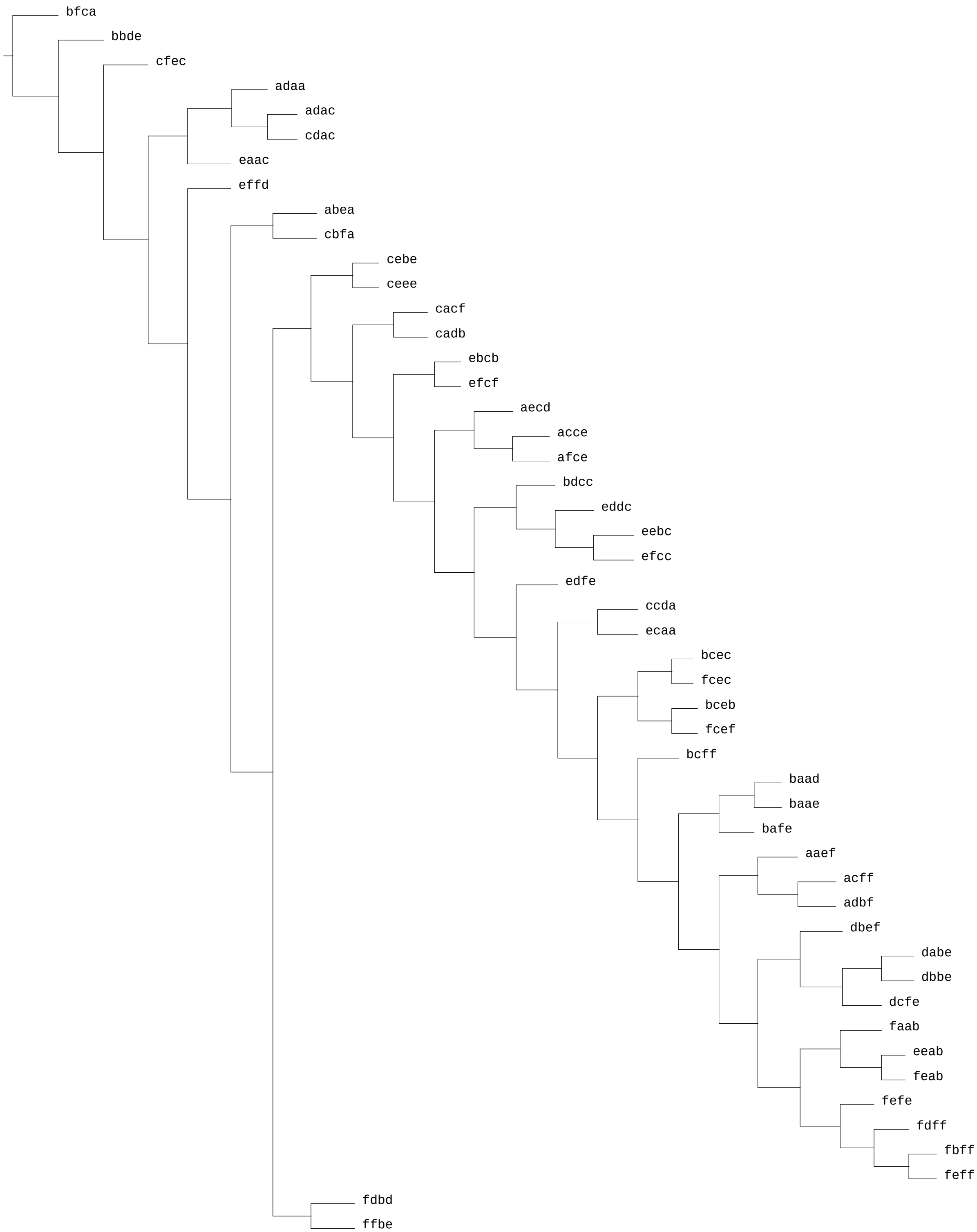


Figure 7.13: Clustering tree of hidden activity patterns corresponding to training input patterns.

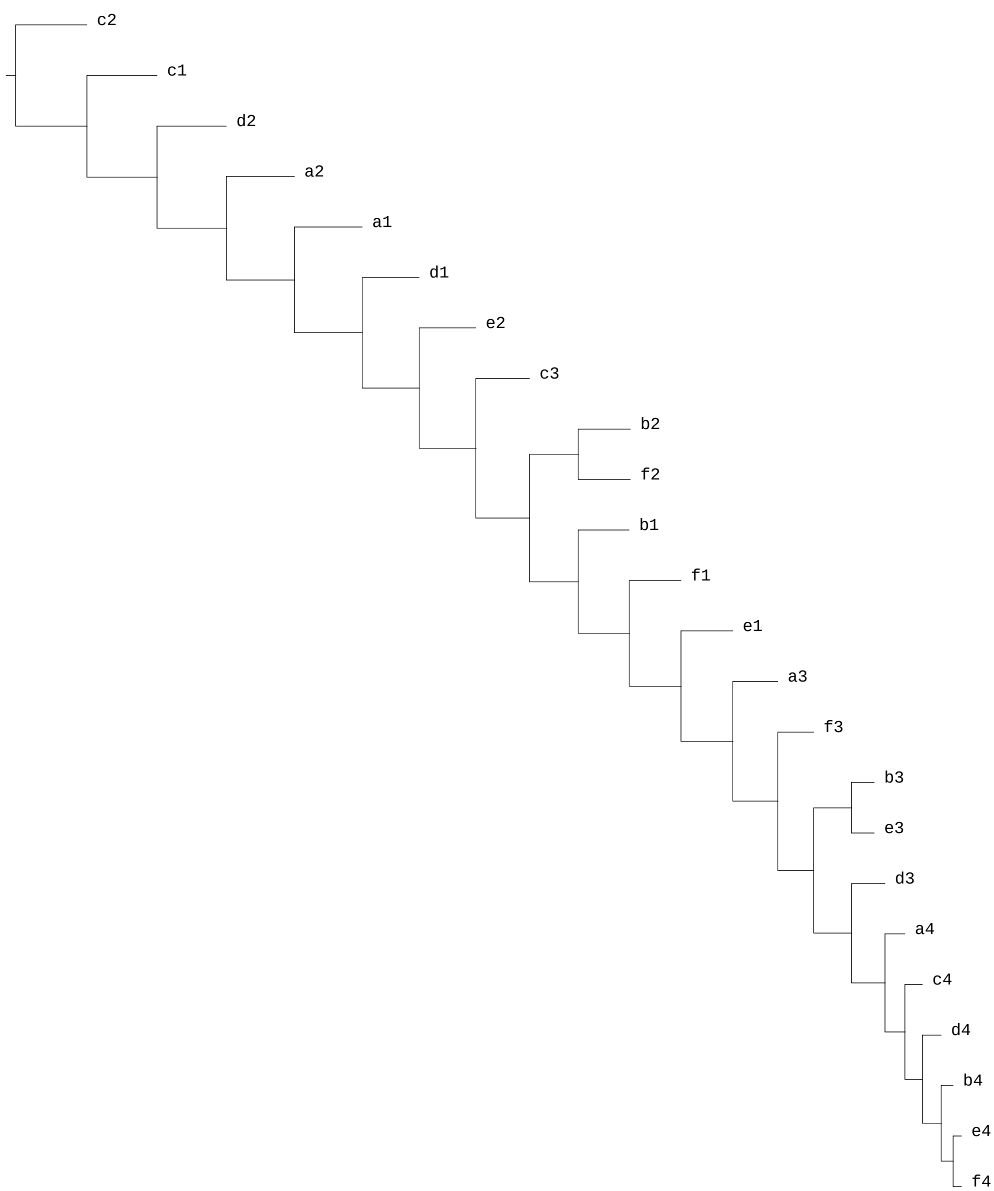


Figure 7.14: Clustering tree of averaged hidden activity patterns corresponding to all patterns of the domain, and averaged by letter/position.

7.6.2 Principal component analysis

Figure 7.15 shows the result of principal component analysis performed on the vectors of hidden unit activities corresponding to the patterns of the training set. We observe that the vectors (labeled by their corresponding input pattern names) are particularly "unclustered", although vectors corresponding to patterns with similar substrings do seem to merge together.

Figure 7.16 shows the result of principal component analysis performed on the same averaged vectors of hidden unit activities which were used earlier for clustering analysis. The figure does not seem to admit any clear interpretation.

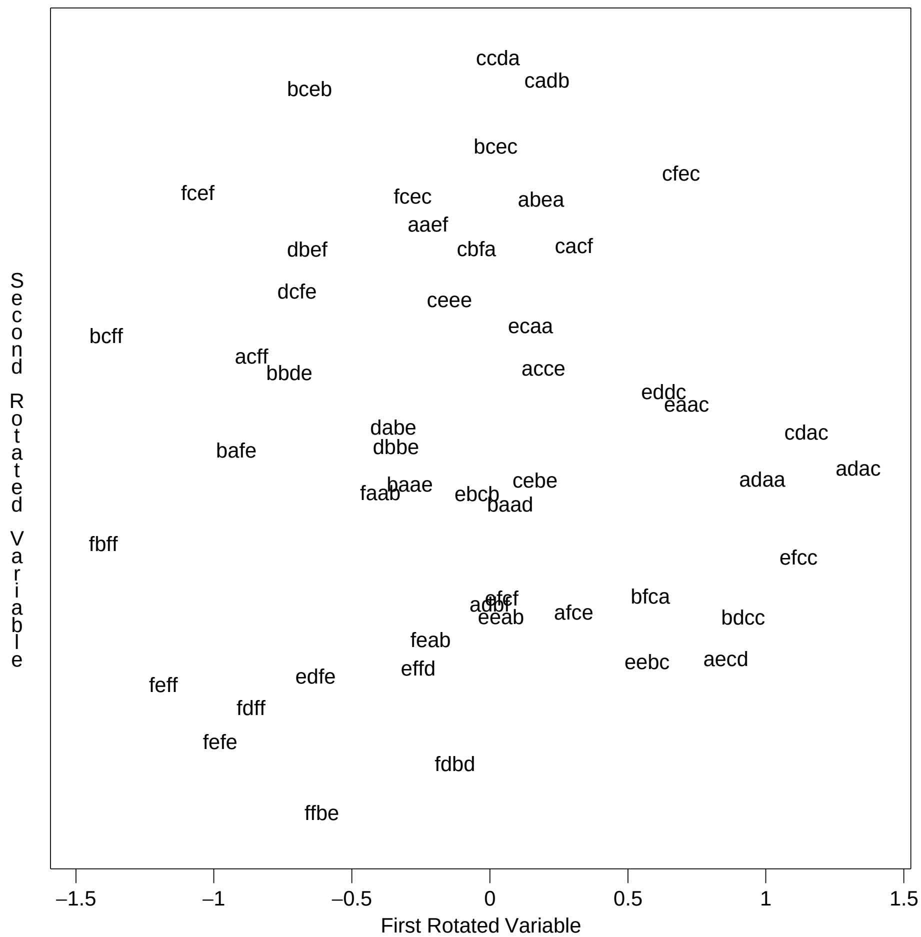


Figure 7.15: Principal component analysis (first two rotated variables) of hidden activity patterns corresponding to training input patterns.

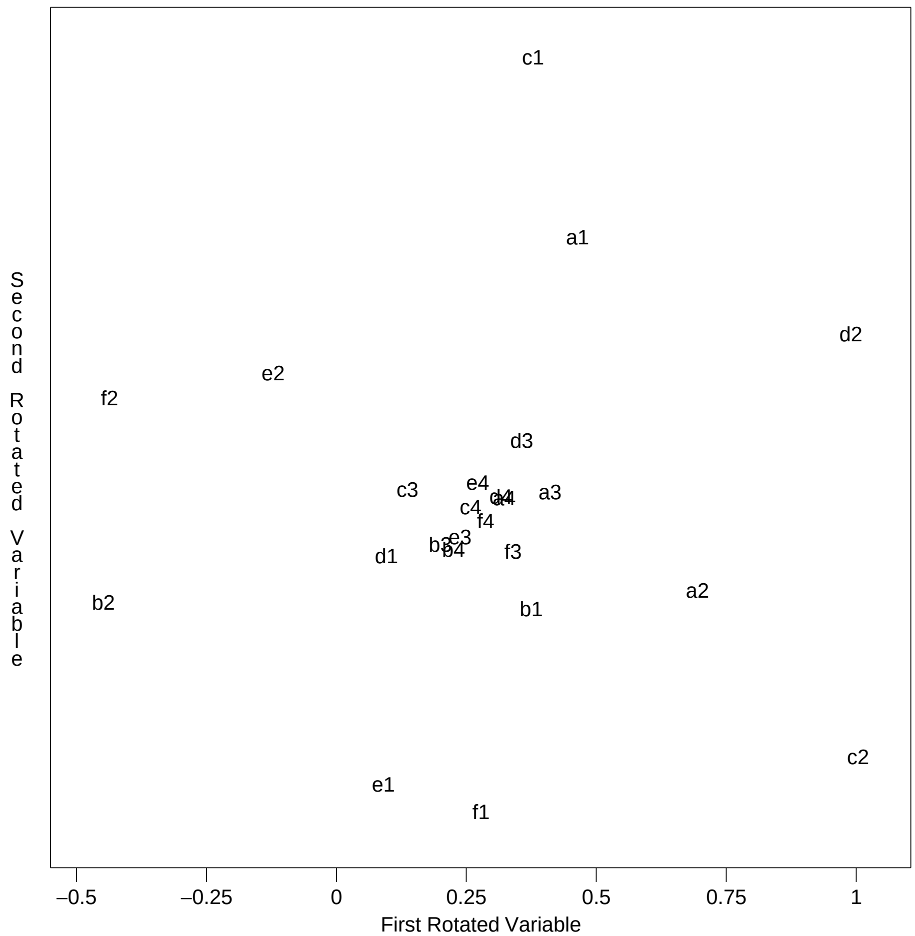


Figure 7.16: Principal component analysis (first two rotated variables) of all hidden activity patterns corresponding to all patterns of the domain, averaged by letter/position.

7.7 Contribution analysis

In this section we report on results obtained with contribution analysis, which will consist in performing principal component analysis on cross-sections of the three-dimensional array of contributions defined for a specific input pattern, a specific output unit, and a specific hidden unit.

The two types of cross-sections on which principal component analysis can be performed, if one is interested in the behavior of the hidden units, are:

- Cross-sections yielding the set of vectors corresponding to the contributions from all hidden units to a specific output unit, for each input pattern presentation. In this case, the principal components correspond to the patterns of hidden units that are responsible for activating the specific output unit, for each input pattern presentation. The responsibilities at play in this case have been called distributed hidden-unit responsibilities by Sanger (1990).
- Cross-sections yielding the set of vectors corresponding to the contributions from a given hidden unit to all output units, for each input pattern presentation. In that case, the principal components correspond to the patterns of output units that the hidden unit is responsible for. The responsibilities at play in this case have been called local hidden-unit responsibilities.

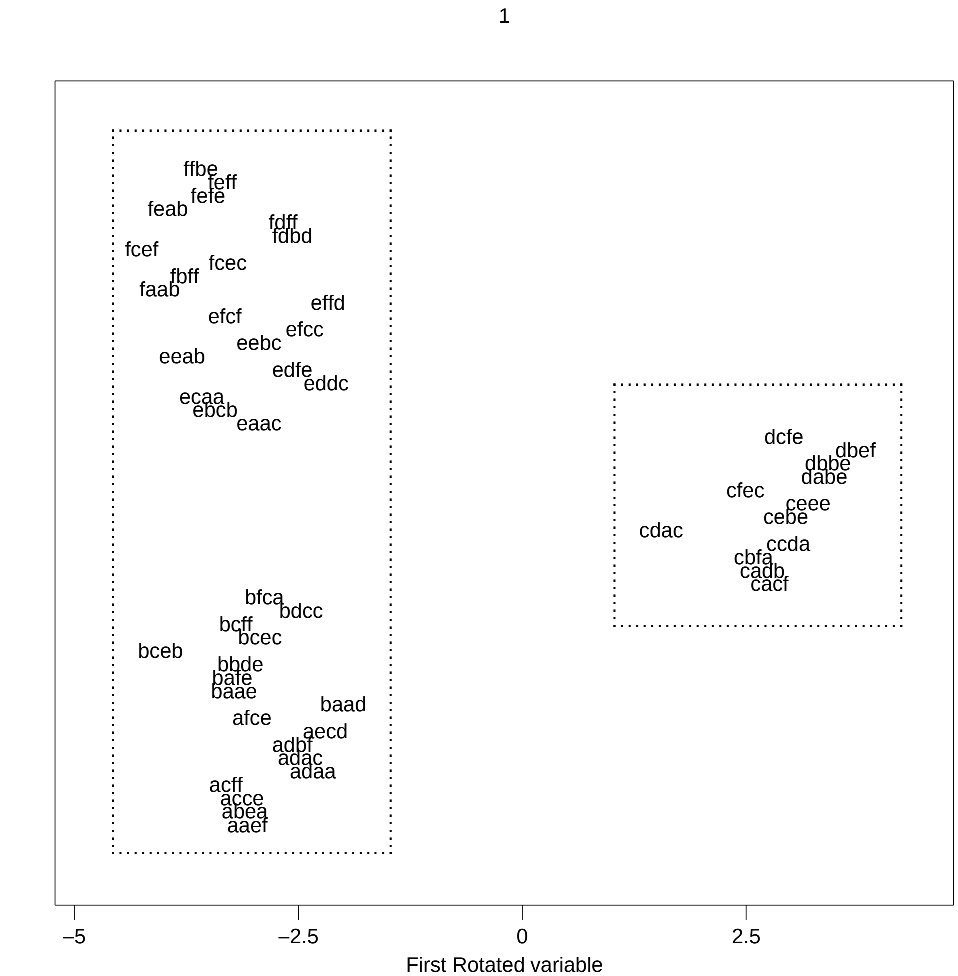
If the network is implementing a loose vertical weight decomposition, the first approach should reveal it: the set of hidden units most responsible for activating an output unit corresponding to a letter in a given position should be most responsible for all (or most) input patterns containing that letter in the same position and any other in the other positions.

7.7.1 Distributed hidden-unit responsibilities

Figure 7.17 shows the results of distributed hidden-unit analysis for the first output unit, participating in the coding of letters in the first position of the strings. The upper graph shows the values of the first rotated variable corresponding to each input pattern of the training set presentation. (The vertical axis does not correspond to any quantity, and points are equally spaced for clarity). We observe a clear vertical division between those patterns of the training set representing strings starting with an "a", "b", "e" or "f", for which output unit is active, and those starting with a "c" or "d", for which output unit 1 is inactive (The axes can, and have been here, be rotated in such a way that a negative value corresponds to correctness).

The graph at the bottom of the figure shows the principal component vector, and therefore yields the pattern of hidden units that are responsible for the output unit being turned on. The pattern is rather distributed.

Figure 7.18 shows similar results for the third output unit, participating in the coding of letters also in the first position. A clear vertical division, again, depending on the first letter of the strings corresponding to the patterns of the training set, is observed.



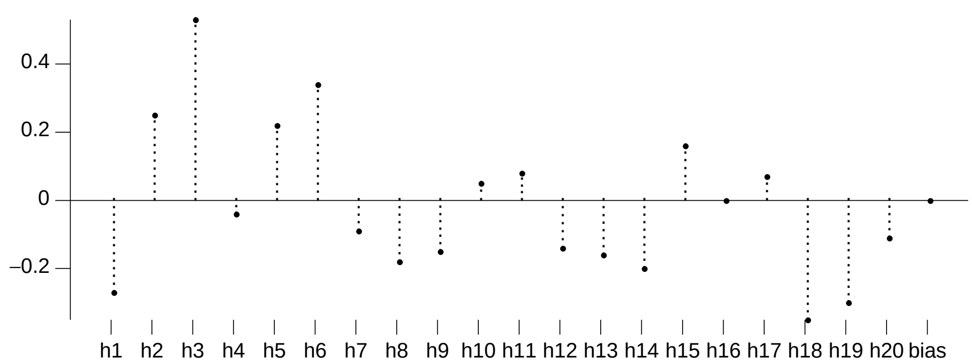


Figure 7.17: Result of contribution analysis for the first output unit: the points correspond to the first rotated variable corresponding to each input pattern presentation. The first principal component is shown at the bottom of the figure.

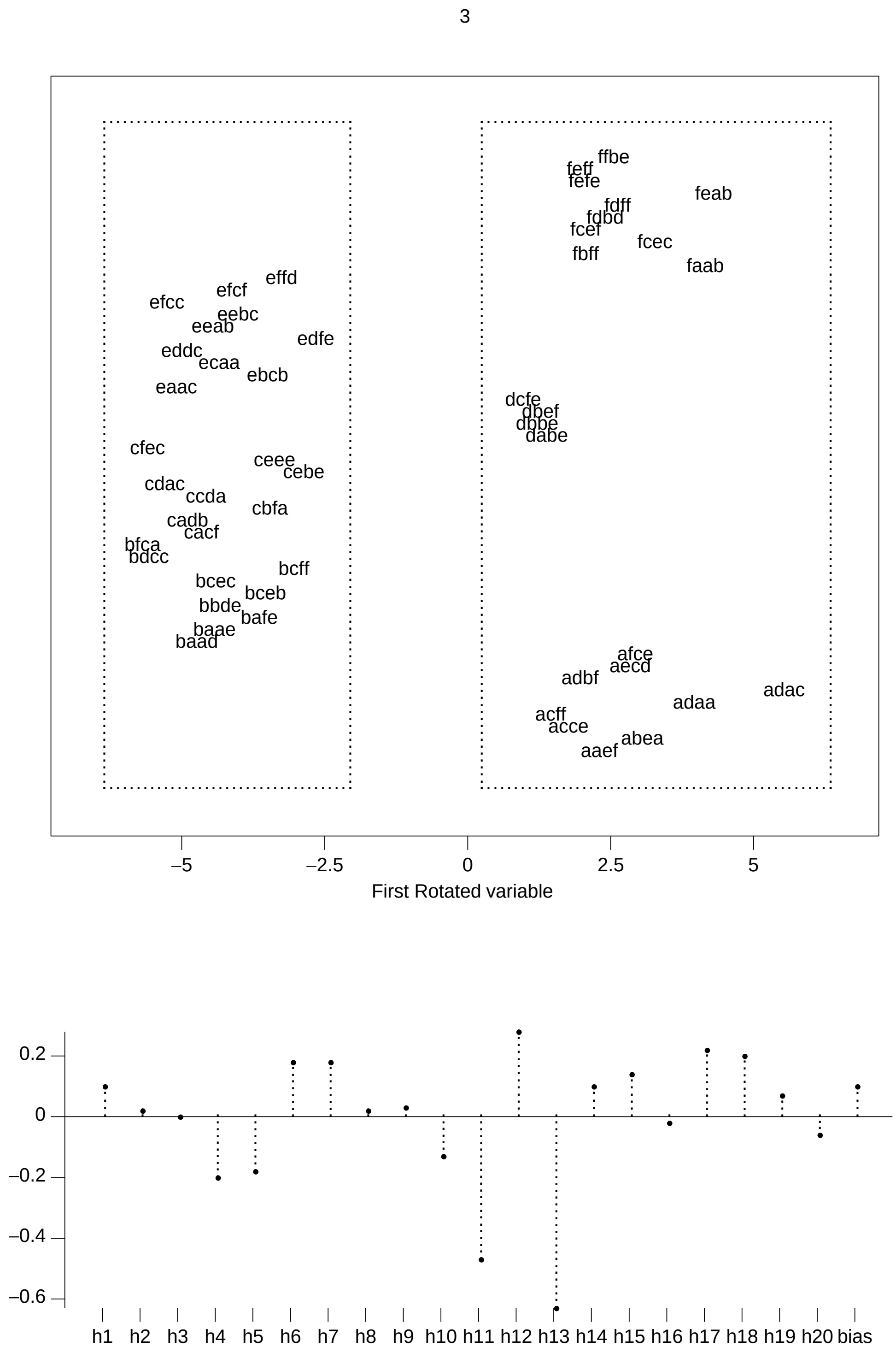


Figure 7.18: Result of contribution analysis for third output unit: Values of the first rotated variable for each input pattern presentation.

Most of the graphs corresponding to each output unit were similar to the two displayed here. Given an output unit, the pattern of hidden units most responsible for that unit were the patterns corresponding to all (or most) input patterns having in the letter position corresponding to the output unit a letter with the corresponding bit on.

Table 7.2 summarizes contribution information, by displaying in a regular expression form the input patterns corresponding to the pattern of hidden units most responsible for a given output unit [5]. The double lines divide the output units in four groups corresponding to constituents of the strings.

We observe what corresponds to a "loose" vertical weight decomposition, where most but not all letters are auto-associated individually regardless of their context, as the input patterns corresponding to hidden unit patterns responsible for an output unit in a given constituent slot have a letter in that given slot. When the table is produced for a similar network trained with weight eliminations (table 7.3 shows hidden unit patterns for one of the networks discussed in section 6.3, where n=4 and $A = 15$), we observe that all letters are auto-associated individually regardless of their context. This analysis thus confirms that, although a first glance at the weights revealed no structure nor trace of vertical decomposition, the network did perform an approximate of such a decomposition.

7.7.2 Local hidden unit responsibilities

Contribution analysis concerning local hidden unit responsibilities was also performed, but was less informative. The interested reader is referred to section D.2 of appendix D for the corresponding study.

[5] A table providing the complete listing for the first 20 output units can be found in section D.1 of appendix D. All related tables were produced automatically by the contribution analysis software package developed by (Sanger, 1990).

Table 7.2: Table of hidden unit patterns.

Output unit	Hidden unit	Pattern presentation
1	h3	c*** (8/8) d*** (4/4)
2	h8 h5	e*c* (3/3)
3	h12	f*** (10/10) a*** (9/9) d*** (4/4)
4	h6 h2	a*** (9/9) c*** (8/8) d*** (4/4)
5	h13 h14 h4 h7	b**c (2/2) *ce* (3/4)
6	h8 h1 h18 h15	e*** (10/10) f*** (10/10) d*** (4/4)
7	h12 h9 h11 h16 h6 h2	a*** (9/9) c*** (8/8)
8	h5	e*** (10/10) c*** (8/8) d*** (4/4)
9	h16 h4	*c** (10/10) *d** (9/9)
10	h8 h5 h9	*fc* (4/4) e*c* (3/3) **c* (8/9) b*** (5/9)
11	h18 h3 h2 h20	*ce* (4/4) *c*f (3/3) *c** (9/10) *b** (5/7)
12	h5 h9 h8	*e** (8/8) *f** (7/7) *b** (7/7)
13	h7 h16 h6 h10	*d** (9/9) *e*e (3/3) *e** (7/8)
14	h7 h10 h9	*d** (9/9) *e** (8/8) *f** (7/7)
15	h1 h19	*c** (10/10) *a** (9/9)
16	h17 h5	*a** (9/9) *f** (7/7) *b** (7/7)
17	h11 h13 h12 h8	**c* (9/9) **d* (4/4)
18	h15	*da* (3/3)
19	h5 h19 h18	***a (2/6)
20	h13 h11	**a* (10/10) **c* (9/9) **d* (4/4)
21	h15 h5 h8 h1 h7 h16 h12	unaccounted for
22	h7 h19 h15 h3 h18 h5 h11 h16	unaccounted for
23	h13	**a* (10/10) **c* (9/9)
24	h18 h19 h2	**c* (9/9) **e* (9/9) **d* (4/4)
25	h10 h11	***c (10/10) ***d (4/4)
26	h1	f*f* (4/4) *a*e (3/3) *df* (2/2) f*b* (2/2) b*f* (2/2) ***d (3/4)
27	h20	***f (11/11) ***a (6/6) ***d (4/4)
28	h11	***c (10/10) ***a (6/6) ***d (4/4)
29	h13 h14 h18 h6	*cec (2/2)
30	h13 h11 h6 h18	***c (10/10) ***a (6/6) ***b (6/6)
31	h11	***c (10/10) ***a (6/6)
32	h6 h20	***f (11/11) ***a (6/6) ***b (6/6)

Table 7.3: Table of hidden unit patterns obtained with a network trained with weight elimination.

Output unit	Hidden unit	Pattern presentation
1	h13 h7	j*** (10/10)
2	h7	h*** (12/12)
3	h24	h*** (12/12) a*** (10/10)
4	h7	h*** (12/12) j*** (10/10)
5	h1	unaccounted for
6	h7	h*** (12/12)
7	h23	b*** (9/9) i*** (9/9) o*** (9/9)
8	h23	unaccounted for
9	h19	*d** (12/12)
10	h21	unaccounted for
11	h22	unaccounted for
12	h19	*d** (12/12)
13	h2	*f** (11/11)
14	h1	unaccounted for
15	h2	*e** (13/13)
16	h3	*f** (11/11)
17	h1	**j* (12/12)
18	h1	**j* (12/12) **h* (9/9) **g* (8/8)
19	h18	**a* (12/12)
20	h13	**j* (12/12)
21	h1	**a* (12/12)
22	h12 h9	**j* (12/12)
23	h9	**i* (9/9) **d* (7/7) **f* (6/6)
24	h12	unaccounted for
25	h17	***j (13/13)
26	h6	***j (13/13) ***g (12/12) ***o (12/12)
27	h20	unaccounted
28	h5	***f (9/9) ***n (8/8)
29	h8	***o (12/12)
30	h17	***j (13/13) ***o (12/12)
31	h17	***o (12/12)
32	h16	***j (13/13)

7.8 Conclusion

While cluster and principal component analysis of the hidden units fell short of providing some information on how the network performed, contribution analysis shed some light, as it showed that a solution close to independent auto-association of each sub-pattern corresponding to a letter (vertical decomposition), for a given position, was implemented. The failure of cluster and principal component analysis might not, after all, be so surprising: Both cluster and principal component analysis use metrics that seem ill-suited at capturing the kinds of regularities induced by the use of semi-distributed tensor product representations in combinatorial domains.

Systematicity and generativity are thus obtained, when the networks are not trained with weights elimination, through approximate weights decomposition.

Chapter 8

Conclusion

8.1 Recapitulation

The success of connectionism as a sub-symbolic theory of cognition is dependent for a great part, we believe, in its ability to explain conceptually-based, or symbolic, behavior. The following question has therefore bee addressed: Since some of the most important and characteristic properties of conceptually-based behavior are generativity and systematicity, and since such properties arise from the fact that structured representations are compositionally constructed from lower level ones and can freely be recombined in systematic yet novel ways, can connectionist models using compositional representations learn to display such properties?

The philosophy underlying our methodology was one of simplicity, as the question above was translated to the following question: If a network is presented with examples of the simplest domain that are combinations (in the simplest sense) of simple constituents, can the most general and least idiosyncratic network learn to exhibit generativity and systematicity in the simple task, massively generalizing from a very small set of examples? The answer to this question was not, we believe, obvious, as generalizations reported in the connectionist literature were typically much smaller than the number of examples used in training sets. And a negative one would have seriously threatened any hope that connectionism can be a general theory of cognition. We hope to have answered it positively, however, by showing that massive generalization can be obtained, as networks learn to recognize the constituent structure of the structured connectionist representations they are trained on. The study of generalization, both from a connectionist and a more abstract theoretical perspective, lead to the concept of virtual generalization, an extension to the notion of generalization that involves very slight learning and which led us to the problem of interference in learning, a hard problem for connectionist models as recent research has shown. More pronounced generativity and systematicity was obtained with virtual generalizations.

The simplicity of the task studied allowed us to interpret the behavior of the networks in an easy way, especially within the context of Cussins' connectionist construction of concept theory: To be generative and systematic, networks need to reduce the perspective dependence of the example they are trained on, which is due to the non-conceptual nature of their connectionist representations. Internal units, allowing for the formation of a cognitive map, allow for such a reduction. When weight elimination is performed, such a reduction can be complete. When it is not, or when the combinatorial complexity of the domain is too high with respect to the size of the training sets used, such a reduction is incomplete, as results from contribution analysis suggested. The resulting network implements a solution which can be interpreted as falling in between the perfect solution equivalent to the implementation of a symbolic system and a solution consisting in pure rote learning of the training set with disregard for its combinatorial structure. Such a result can be seen as an answer to the arguments raised in (Fodor and Pylyshyn, 1988) and (Fodor and McLaulghlin, 1990) claiming that connectionist systems would have to implement a symbolic system to display generativity and systematicity.

8.2 Shortcomings

The limitations of this thesis are numerous. While the simple approach taken allowed a clear interpretation, and was thus preferable, in our opinion, to a more complex, richer but potentially uninterpretable one consisting for instance in building a psychologically plausible model for a non-obvious cognitive task involving generativity and systematicity, it left an number of important questions unanswered: can networks trained on domains which are not as obviously or as simply combinatorial exhibit the same kind of properties? Can the results reported in this thesis translate to other types of learning algorithms or network architectures? Experiments reported on the words domain suggest that the answer to the first question is yes. And the simplicity of the learning algorithm and architecture used intimate that the answer to the second question is also positive.

Our characterization of combinatorial domains, and what were called structure preserving connectionist representational schemes, while unearthing a number of important issues, felt short of being complete and formal.

Within the task studied, a number of limitations can be put forward. The concept of virtual generalization, and the related problem of interference, was only studied in its simplest form: Only one untrained example was presented to the networks having learned the domain, and tested for interference. A study of interference, or lack thereof, produced by multiple untrained pattern could have shed more light on the subject.

8.3 Impact of this thesis on some related issues

8.3.1 The empiricism and nativism debate

Ironically, as a model of human learning, connectionism has tended to go in disagreement with the empiricist position in the empiricist/nativist debate, as lengthy training regimes (in the sense that a large fraction of the domain needed to be presented) seemed to be the rule. The results reported in this thesis, however, suggest that in combinatorial domains, such training regimes are uncalled for. Although we have not provided evidence that connectionist induction algorithms are stronger than previously available inductive techniques, we do believe that the results obtained in this thesis suggest that connectionism is more compatible with an empiricist position on human learning than previous results would suggest–at least within combinatorial domains.

8.3.2 The learning and processing dualism

Learning (in the sense of synaptic adjustment), and processing (in the sense of activity flow only) have traditionally been regarded as two distinct aspects of human cognition, usually separated in time. Traditional memory experiments, for instance, are typically divided into two distinct phase, one of learning and memorizing, and the other of remembering and processing. The very small amount of weight adjustment required to learn a virtual generalization which would not interfere with previously learned models could suggest that the learning and processing dualism might not always be the rule, as there is, after all, no reason to believe that even the simplest processing task might not involve small, rapid, and interference-free relearning (indeed, instances of synaptic weights adjustments have been found in very short time-frames, e.g. (Edelman, 1987)).

8.4 Future directions

As was mentioned earlier, a study of how multiple patterns learned on top of a network having learned the domain would interfere would be of interest. Keeping the same architecture, a study of more complex cognitively related tasks, such as pattern association (in contrast to auto-association) could shed a better light on more complex instances of systematic or generative cognitive behavior. A further development of the formal study of combinatorial domains would also be most valuable.

Another enterprise would consist in building a proper memory model, allowing possibly for temporal processing. The additional time dimension in such processing introduces the possibility for much richer structure, as domains ranging from regular languages to context-free or context sensitive languages could be studied.

Appendix A

Conditional entropy inequality

An analytical proof of the inequality 3.2, based on (Ross, 1988), is here given.

$$E_{A_1}(A_2) - E(A_2)$$

$$= -\sum_{a_1} P(a_1) \sum_{a_2} P_{a_1}(a_2) \log_2 P_{a_1}(a_2) + \sum_{a_2} P(a_2) \log_2 P(a_2)$$

$$= -\sum_{a_1} \sum_{a_2} P(a_1, a_2) \log_2 P_{a_1}(a_2) + \sum_{a_2} \{ \sum_{a_1} P(a_1, a_2) \} \log_2 P(a_2)$$

$$= \sum_{a_1} \sum_{a_2} P(a_1, a_2) \log_2 \frac{P(a_2)}{P_{a_1}(a_2)}$$

$$\leq \log_2(e) \ \sum_{a_1} \sum_{a_2} P(a_1, a_2) [\frac{P(a_2)}{P_{a_1}(a_2)} - 1]$$

(since $\forall x > 0$, $\ln x \leq x - 1$)

$$= \log_2(e) \ \sum_{a_1} \sum_{a_2} P(a_1) P_{a_1}(a_2) [\frac{P(a_2)}{P_{a_1}(a_2)} - 1]$$

$$= \log_2 e \ \ [\sum_{a_1} \sum_{a_2} P(a_1) P(a_2) - \sum_{a_1} \sum_{a_2} P(a_1, a_2)]$$

$$= 0$$

Appendix B

Unbinding

The proof that there exists an unbinding vector u_i recovering each filler f_i from

$$D_i = \sum_{i=1}^{P} f_i \otimes r_i$$

when the role vectors r_i are linearly independent, can here be summarized as follows:
If the roles r_i are linearly independent, then they form a basis B of the vector space V that they span. The linear functions r_i^* that form the dual basis B^* of the dual space V^* of V, have the property:

$$\forall (i,j) \in \{1, \ldots, P\}^2, r_i^*(r_j) = \delta_{ij}$$

where δ is the Kronecker symbol. If u_i is the unique vector of the axis orthogonal to the hyperplane spanned by all vectors of B save r_i such that $r_i.u_i = 1$, then

$$\forall x \in V, r_i^*(x) = x.u_i$$

This is because, if x_i are the coordinates of x in B, then

$$r_i^*(x) = r_i^*(\sum_{j=1}^{P} x_j r_j)$$

$$= \sum_{j=1}^{P} r_i^*(x_j r_j)$$

$$= x_i$$

$$= x_i(r_i.u_i) \quad (\text{since } r_i.u_i = 1)$$

$$= (\sum_{j=1}^{P} x_j r_j).u_i \text{ (since } u_i \text{ is orthogonal to all vectors of B save } r_i)$$

$$= x.u_i$$

It follows that:

$$(\sum_{j=1}^{P} f_j \otimes r_j).u_i$$

$$= \sum_{j=1}^{P} f_j(r_j.u_i) = \sum_{j=1}^{P} f_i\delta_{ij} = f_i$$

Appendix C

A naive analysis

C.1 Introduction

The task for the network is to learn the combinatorial domain X consisting of sequences $a_1a_2..a_n$ where n defines the length of the sequence, and a_i can take any value of the associated alphabet A. The network is assumed to be linear and the sequences $a_1a_2..a_n$ are mapped to a vector according to the tensor product representation:

$$a_1a_2...a_n \rightarrow \sum_{i=1}^{n} a_i \otimes r_i$$

where r_i's are distinct vectors representing the position of a_i in the sequence, and a_i's are vectors representing members of the alphabet A.

The property mentioned in footnote 7.2.1 of chapter 7 is:

$$\forall \;\; a_1, a_2, \ldots a_n, x, y, \in A$$

$$\forall \;\; 1 \leq i < j \leq n$$

then [1]

$$a_1a_2...a_{i-1}xa_{i+1}...a_{j-1}ya_{j+1}...a_n$$

$$= \; a_1a_2...a_{i-1}xa_{i+1}...a_n \; + \; a_1a_2...a_{j-1}ya_{j+1}...a_n \; - \; a_1a_2...a_n$$

[1] Simply:

$$a_1a_2..x..y..a_n = a_1a_2..x..x_n + a_1a_2..y..a_n - a_1a_2..a_n$$

Any sequence of n letters can thus be decomposed in three sequences. For sequences of length 2, for instances, the vector representing ad can be rewritten as the sum of vectors ab + cd − cb.

With this property, it is possible to derive that, in a linear auto-associative network, if:

- All the vectors a_i's representing members of A are orthogonal and normalized,
- All role vectors r_i are orthogonal and normalized,
- The training set is learned with the delta rule within an error criterion ϵ and generalizations and virtual memories are tested with an error criterion $5/2\epsilon$,
- $a_1a_2..x..a_n$, $a_1a_2..y..a_n$ and $a_1a_2..a_n$ are in the training set

then $a_1a_2..x..y..a_n$ is either a true or virtual generalization.

Proof: We here investigate the case for $n = 2$. If the weight matrix W has already learned the vectors xa_i, a_jy and a_ja_i to criterion ϵ, then, defining $E = W - I$, we have:

$$Wa_ja_i = a_ja_i + Ea_ja_i$$

$$Wxa_i = xa_i + Exa_i$$

$$Wa_jy = a_jy + Ea_jy$$

$$||\epsilon a_ja_i|| \ \leq \ \epsilon$$

$$||\epsilon xa_i|| \ \leq \ \epsilon$$

$$||\epsilon a_jy|| \ \leq \ \epsilon$$

where || || is the norm on the space of input (or output) vectors.
We have,

$$xy \ = \ xa_i \ + \ a_jy \ - \ a_ja_i$$

Therefore,

$$Wxy \ = \ Wxa_i \ + \ Wa_jy \ - \ Wa_ja_i$$

and

$$Exy \;=\; Exa_i \;+\; Ea_j y \;-\; Ea_j a_i$$

If

$$||Exy|| \;=\; ||Exa_i \;+\; Ea_j y \;-\; Ea_j a_i|| \;\leq\; \frac{5}{2}\epsilon \quad (1)$$

then xy is a generalization. [2]

If not, then we show that xy is a virtual generalization. For this, we will show that:

- xy can be learned in one learning trial.
- xy will cause no interference with the previously learned patterns.

The first condition is verified if, after applying the delta learning rule with a learning rate η, the new matrix W_{new} verifies

$$W_{new} xy \;=\; xy$$

Now, since

$$W_{new} \;=\; W \;-\; \eta\,(Wxy - xy)\; xy^T,$$

a learning rate of

$$\eta \;=\; \frac{1}{(xy)^2} \;=\; \frac{1}{2}$$

verifies (1).

The second condition is verified if the previously learned vectors $a_k a$ are correctly auto-associated by W_{new}.

If xy has no letter in common with the $a_k a$, then W_{new} and W will perform identically on $a_k a$.

If $a_k a$ has one letter in common with xy, say y, then:

$$W_{new} a_k y$$

$$= Wa_k y \;-\; \frac{(Wxy \;-\; xy)\; xy^T}{2}\; a_k y$$

$$= a_k y \;+\; Ea_k y \;-\; (Exa_i \;+\; Ea_j y \;-\; Ea_j a_i)\; \frac{(xy^T a_k y)}{2}$$

[2]Note that $||Exa_i \;+\; Ea_j y \;-\; Ea_j a_i|| \;\leq\; 3\epsilon$.

$$= a_k y + E a_k y - \frac{(E x a_i + E a_j y - E a_j a_i)}{2}$$

$$= a_k y + \frac{2 E a_k y - E x a_i - E a_j y + E a_j a_i}{2}$$

Since

$$|| \frac{2 E a_k y - E x a_i - E a_j y + E a_j a_i}{2} || \leq \frac{5}{2} \epsilon \quad (2)$$

($||E a_k y|| \leq \epsilon$, $||E x a_i|| \leq \epsilon$, $||E a_j y|| \leq \epsilon$, $||E a_j a_i|| \leq \epsilon$), $a_k y$ correctly generalizes on W_{new}.
By symmetry, the same conclusion holds for xa.

C.2 Remarks

- A new sequence will not interfere with a previously learned one that has no common letters.
- The term Exy can furthermore be decomposed in the sum of two terms which depend only on x and y, respectively.

$$W x y = x y + E x y$$

$$= x y + E x_1 + E_2 y$$

where

$$E_1 = (W - I) \otimes r_1$$

$$E_2 = (W - I) \otimes r_2$$

Condition (1) for generalization is ensured by, using now the same error criterion ϵ for testing and training:

$$||E_1 x|| + ||E_2 y|| \leq \epsilon$$

Condition (2) for non-interference is ensured by:

$$||\frac{2E_1a_k \ + \ E_2y \ - \ E_1x}{2}|| \ \leq \ \epsilon$$

We might be tempted to conjecture that, given the symmetry of the problem, when a letter x is in a sequence that has been correctly learned by the network, then

$$||E_ix|| \ , \ i = 1, 2$$

is close to $\frac{1}{2}\epsilon$

In that case (1) and (2) hold.

- In the more general case where vectors representing a_i's are neither orthogonal nor normalized, and the same error criterion is used for training and testing, then condition (2) is:

$$||a_ka_l \ + \ cos(a_ka_l, \ xy)\frac{||a_ka_l||}{||xy||}(Exa_i \ - \ Ea_jy \ + \ Ea_ja_i)|| \ \leq \ \epsilon \quad (3)$$

  (xy might interfere with a previously learned pattern that has no letter in common).

Condition (3) will diminish the number of virtual generalizations and generalizations, in a way that solely depends on the representation used, by a constant, that is, since the same representation is used for all experiments.

C.3 Verification

Although the conditions for such a property are severe, we tested what effects property (1) might have on the number of generalizations and virtual generalizations in our model network, which could account in part for the numbers found experimentally, in the more complex networks.

This was done by calculating the probability that a pattern xy in the testing set has x in common with a pattern xa_i in the training set, y in common with a pattern a_jy in the training set, and that a_ja_i is in the training set. An estimation of the number of generalizations and virtual generalizations was then possible. This naive approach, unfortunately but not surprisingly, did not prove to be conclusive, as the true and virtual generaliza-

Appendix D

Further contribution analysis

D.1 Full table of hidden units patterns

Table D.1 list all input patterns corresponding to the pattern of hidden units most responsible for a given output (first 20 output units), for the experiment analyzed in section 7.7 of chapter 7.

Table D.1: Full table of hidden unit responsibilities

Output unit	Hidden unit	Pattern presentation
1	h3	dbef dbbe dabe ceee ccda cebe dcfe cacf cadb cbfa cfec cdac
2	h8 h5	ebcb efcf efcc ffbe
3	h12	adac feab faab adaa fcec abea afce aecd fdff ffbe fdbd aaef fcef fbff fefe feff adbf acce acff dabe dbef dbbe dcfe
4	h6 h2	acff cfec aaef dcfe cadb dabe ccda adbf adac ceee cacf cdac dbef abea cbfa dbbe adaa cebe afce acce aecd
5	h13 h14 h4 h7	bdcc bcec fcec bceb feab
6	h8 h1 h18 h15	effd ffbe fdbd fefe eebc fbff eaac efcc feff edfe fdff eddc efcf eeab fcec faab fcef ecaa feab dbbe dabe dbef ebcb dcfe
7	h12 h9 h11 h16 h6 h2	acff aaef adbf adac afce abea adaa aecd cacf cfec cadb ccda acce cbfa ceee cdac cebe
8	h5	cebe ceee dabe dbbe cfec efcc eebc cacf cbfa effd eaac ebcb efcf ecaa eeab dbef ccda cdac edfe cadb dcfe eddc
9	h16 h4	fdbd bdcc fdff edfe eddc adbf bcff fcef fcec adac adaa acff cdac bcec bceb ccda ecaa dcfe acce
10	h8 h5 h9	efcf bfca effd afce ffbe efcc ebcb fdbd aecd bdcc bbde bcff edfe acce bafe
11	h18 h3 h2 h20	ecaa acff bcff fcef ccda feab bceb eeab abea ebcb dbef acce fbff bbde bcec fcec
12	h5 h9 h8	efcf effd ffbe fbff bfca feff cebe afce eebc dbbe dbef efcc ebcb cfec fefe ceee aecd cbfa abea eeab bbde feab
13	h7 h16 h6 h10	fdff adac bdcc adaa fdbd cdac adbf eebc fefe eddc feff edfe feab eeab cebe ceee
14	h7 h10 h9	fefe cebe ceee ffbe feab feff aecd eeab adac eebc fdbd bfca fdff bdcc cfec afce adaa adbf cdac edfe effd efcf efcc eddc
15	h1 h19	bafe baae baad dabe faab bcff aaef eaac fcef acff cacf ecaa fcec bceb dcfe bcec cadb acce ccda
16	h17 h5	ffbe cfec bfca abea efcc eaac faab fbff dbbe dabe bafe cbfa afce effd aaef bbde dbef baad baae ebcb efcf cadb cacf
17	h11 h13 h12 h8	cadb eddc bfca efcc cacf efcf bdcc ccda bbde afce ebcb aecd acce
18	h15	adac cdac adaa
19	h5 h19 h18	abea bfca
20	h13 h11	ccda bbde cadb bfca eddc cacf bdcc efcc eaac ecaa baae baad eeab efcf cdac adac faab feab ebcb adaa afce aecd acce

D.2 Local hidden unit responsibilities

Figure D.1 and D.2 are the result of principal component analysis (first two principal components) of contributions, for the first and third hidden units, respectively. The corresponding experiment is described in section 7.7 of chapter 7.

Table D.2 summarizes contribution information for each hidden unit, by displaying in a regular expression form the input patterns most responsible for a given hidden unit.

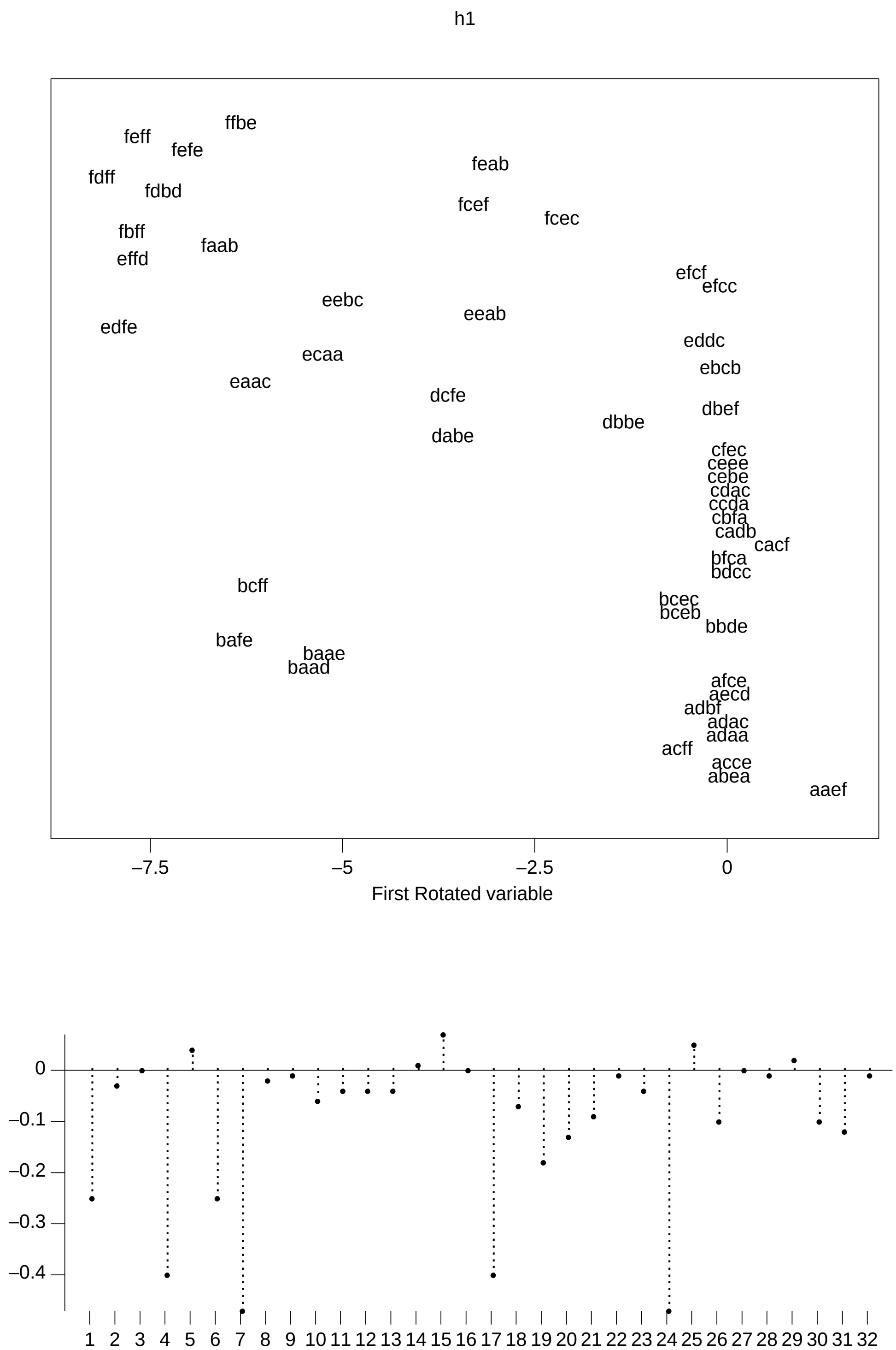


Figure D.1: Principal component analysis (first two principal components) of contributions, for first hidden unit

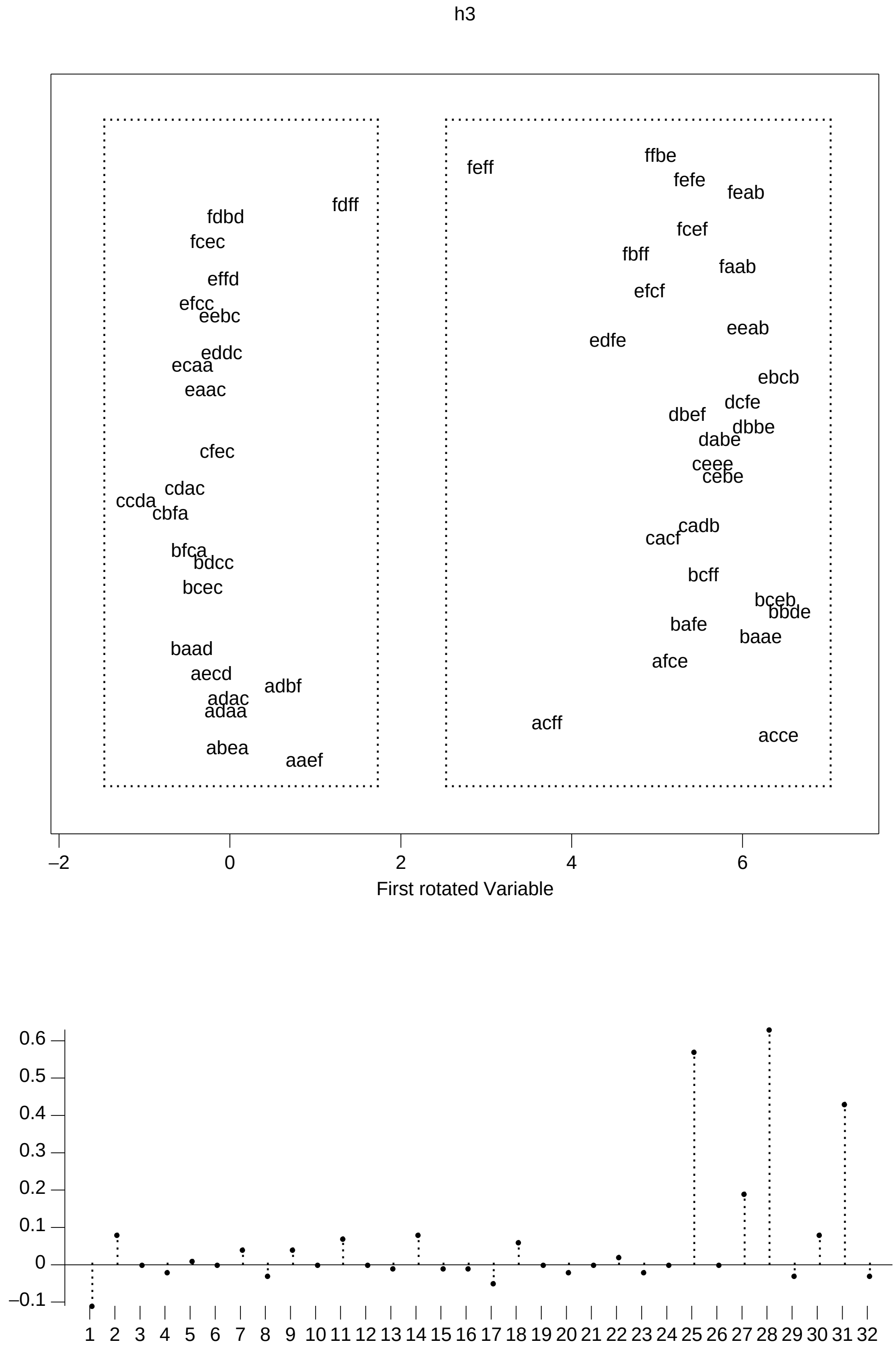


Figure D.2: Principal component analysis (first two principal components) of contributions, for third hidden unit.

Table D.2: Table of hidden unit responsibilities

Hidden unit	Output Unit	Input presentations
h1	7 24 17 4	f*f* (4/4) *aa* (4/4) **fe (3/4) e*** (5/10)
h2	1	not accounted for
h3	1	***c (10/10) ***a (6/6) ***d (4/4)
h4	22 23	f*f* (4/4) *ce* (4/4) **fe (4/4) c*** (4/8)
h5	19 12 9	e*c* (3/3) *eb* (2/2) db** (2/2) *f** (5/7)
h6	21	**e* (9/9) **a* (9/10)
h7	13 14 16 15	*e** (8/8) ad** (3/3) *d** (7/9)
h8	6	f*f* (4/4) ef** (3/3) fe** (3/3) e**b (2/2) e*** (7/10)
h9	7	a**a (2/2) c*** (2/8)
h10	27 32	*d*c (4/4) *e*e (3/3) ***c (6/10)
h11	28	*d*c (4/4) e**c (4/4) ***a (5/6)
h12	3 7	a*** (9/9)
h13	3	b*** (9/9) e**c (4/4) e*** (9/10) c*** (5/8)
h14	29 8 5	f**f (4/4) *ce* (4/4) f**b (2/2) b*** (4/9)
h15	21 24 29 32 6	**be (4/4) e*f* (2/2) f**e (2/2) **b* (6/7)
h16	13 11 12	*a** (9/9) *d** (9/9)
h17	16	*aa* (4/4) *a*e (3/3)
h18	11	*ce* (4/4)
	19 5 24 11 13	not accounted for
	29 6 1 11 7 30 5 4	e**c (4/4) *ce* (4/4) e*a* (3/3) e*c* (3/3)
	24	**c* (2/9)
	16	a*c* (2/3)
h19	8	*ce* (4/4) b**e (3/3) a*c* (3/3) b*** (8/9)
	19 24	*ce* (4/4)
	14 15 8	ba** (3/3)
h20	19	*cff (2/2) c**a (2/2) **ff (4/5) ***a (4/6)
	27 32	not accounted for
bias	3	f*** (10/10) a*** (9/9) d*** (4/4)

Bibliography

(Abu-Mostafa, 1986) Yaser S. Abu-Mostafa. The complexity of information extraction. *IEEE Transactions on Information Theory*, IT-32:513–525, 1986.

(Ackley *et al.*, 1985) D.H. Ackley, G.E. Hinton, and T.J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

(Ahmad, 1988) Subutai Ahmad. A study of scaling and generalization in neural networks. Technical Report UIUCDCS-R-88-1454, Department of Computer Science, University of Illinois at Urbana-Champaign, September 1988.

(Almeida, 1987) L.B. Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Proceedings of the IEEE First International Conference on Neural Networks, San Diego*, pages 609–618. New York: IEEE, 1987.

(Anderson *et al.*, 1977) J.A. Anderson, J.W. Silverstein, S.A. Ritz, and R.S. Jones. Distinctive features, categorical perception and probability learning: Some applications of a neural model. *Psychological Review*, 84:413–451, 1977.

(Attneave, 1959) Fred Attneave. *Applications of information theory to psychology: A summmary of basic concepts, methods, and results.* New York: Holt, Rinehart and Winston, 1959.

(Baldi and Hornik, 1988) P. Baldi and K. Hornik. Neural networks and principal components analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58, 1988.

(Barnes and Underwood, 1959) J.M. Barnes and B.J. Underwood. Fate of first-list associations in transfer theory. *Journal of Experimental psychology*, 58:97–105, 1959.

(Baum and Haussler, 1989) Eric Baum and David Haussler. What size net gives valid generalization. *Neural Computation*, 1:151–160, 1989.

(Blumer *et al.*, 1987) Anselm Blumer, Andrzej Ehrenfeucht, David Hausler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. Technical Report UCSC-CRL-87-20, University of California, Santa Cruz, 1987.

(Brousse and Smolensky, 1989a) Olivier Brousse and Paul Smolensky. Connectionist generalization and incremental learning in conbinatorial domains. In *Synergetics of Cognition, H. Haken (Ed.)*, pages 126–133, New York, 1989. Springer-Verlag.

(Brousse and Smolensky, 1989b) Olivier Brousse and Paul Smolensky. Virtual memories and massive generalization in connectionist combinatorial learning. In *Proceedings of the Eleventh Annual Cognitive Science Society Conference*, pages 26–33, Hillsdale, NJ, 1989. Lawrence Erlbaum Associates.

(Brousse and Smolensky, 1990) Olivier Brousse and Paul Smolensky. Interference and generalization in connectionist networks: Within-domain structure or between-domain correlation? A response. *Neural Network Review*, 4:27–29, 1990.

(Brousse, 1991) Olivier Brousse. *Generativity and Systematicity in Neural Network Combinatorial Learning.* PhD thesis, University of Colorado, Boulder, 1991.

(Carpenter and Grossberg, 1987) G. A. Carpenter and S. Grossberg. Art2: Self organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26:4919–4930, 1987.

(Chaitin, 1965) G.J. Chaitin. A theory of program size formally identical to information theory. *Journal of the ACM*, 22:329–340, 1965.

(Chalmers, 1990a) David Chalmers. Syntactic transformations on distributed representations. Technical report, Center for research on concepts and cognition, Indiana University, 1990.

(Chalmers, 1990b) David Chalmers. Why Fodor and Pylyshyn were wrong: the simplest refutation. Technical report, Center for research on concepts and cognition, Indiana University, 1990.

(Chauvin, 1990) Yves Chauvin. Generalization performance of overtrained networks. In *Proceedings of the 1990 EURASIP workshop*, page 46. Springer, 1990.

(Chomsky, 1968) N. Chomsky. *Language and mind.* New York: Harcourt, Brace and World, 1968.

(Cover and King, 1978) Thomas M. Cover and Roger C. King. A convergent gambling estimate of the entropy of English. *IEEE Transactions on Information Theory*, IT-24:413–421, 1978.

(Cussins, 1990) Adrian Cussins. The connectionist construction of concepts. In Margaret A. Boden, editor, *Philosophy of artificial intelligence*, pages 368–440. 1990.

(Denker *et al.*, 1987) John Denker, Daniel Schwartz, Ben Wittner, Sara Solla, Richard Howard, Lawrence Jackel, and John Hopfield. Large automatic learning, rule extraction and generalization. *Complex Systems*, 1:877–822, 1987.

(Derthick, 1986) Mark Derthick. Is distributed connectionism compatible with the physical symbol hypothesis ? In *The Eight Conference of the Cognitive Science Society*, pages 639–644, Hillsdale, NJ, 1986. Lawrence Erlbaum Associates.

(Dreyfus and Dreyfus, 1988) H.L. Dreyfus and S.E. Dreyfus. Making a mind versus modeling a brain: Artificial intelligence back at a branchpoint. In *Proceedings of the American Academy of Arts and Sciences*, pages 15–43, Daedalus, P.O. Box 515, Canton, MA 02021, 1988. American Academy of Arts and Sciences.

(Edelman, 1987) Gerald M. Edelman. *Neural Darwinism : the theory of neuronal group selection.* New York : Basic Books, 1987.

(Ehrenfeucht, 1991) Andrzej Ehrenfeucht. Personal Communication, 1991.

(Elman, 1990) J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

(Elman, 1991) Jeffrey L. Elman. Incremental learning, or the importance of starting small. In *Proceedings of the Thirteenth Annual Cognitive Science Society Conference*, Hillsdale, NJ, 1991. Lawrence Erlbaum Associates.

(Estes, 1982) W.K. Estes. Similarity-related channel interactions in visual processing. *Journal of Experimental Psychology: Human Perception and Performance*, 38:353–382, 1982.

(Estes, 1991) W.K. Estes. Cognitive architectures from the standpont of an experimental psychologist. *Annual Review of Psychology*, 42:1–28, 1991.

(Everitt, 1980) Brian Everitt. *Cluster analysis.* Halsted Press, New York, 1980.

(Fodor and McLaulghlin, 1990) J.A. Fodor and B.P. McLaulghlin. Connectionism and the problem of systematicity: Why Smolensky's solution won't work. *Cognition*, 35:183–204, 1990.

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

(Fodor, 1975) J.A. Fodor. *The language of thought.* Harvard University Press, 1975.

(Fozzard *et al.*, 1989) Rich Fozzard, Lou Ceci, and Gary Bradshaw. Theonet, a connectionist expert system that actually works. In D. S. Touretzky, editor, *Advances in neural information processing systems 1*, pages 248–255. Morgan Kauffman, San Mateo, 1989.

(Frean, 1990) M. Frean. The Upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation*, 2:198–209, 1990.

(French, 1991) Robert M. French. Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. Technical Report CRCC 51-1991, Center for research on concepts and cognition, Indiana University, 1991.

(Gardner and Derrida, 1989) E. Gardner and B. Derrida. Three unfinished works on the optimal storage capacity of networks. *Journal of physics A*, 22:1983–1994, 1989.

(Garner, 1962) W.R. Garner. *Uncertainty and Structure as Psychological Concepts.* John Wiley ad Sons, Inc., New York, 1962.

(Goggin *et al.*, Submitted) Shelly D.D. Goggin, Karl E. Gustafson, and Kristina M. Johnson. An analysis of the hidden unit and weights values in nonlinear multilayer neural networks. *IEEE Transaction on Neural Networks*, ?, Submitted.

(Grassberger, 1989) Peter Grassberger. Sequences and efficient codes. *IEEE Transactions on Information Theory*, 35:669–675, 1989.

(Grossberg, 1976) S. Grossberg. Adaptive pattern classification and universal recoding: I. parralel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121–134, 1976.

(Hanson and Kegl, 1987) J.H. Hanson and J. Kegl. Parsnip: A connectionist network that learns natural language grammar from exposure to natural language sentences. In *The Ninth Conference of the Cognitive Science Society*, pages 106–119, Hillsdale, NJ, 1987. Lawrence Erlbaum Associates.

(Hanson and Pratt, 1989) S.J. Hanson and L. Pratt. A comparison of different biases for minimal network constructions with back-propagation. In D. S. Touretzky, editor, *Advances in neural information processing systems 1*, pages 177–185. Morgan Kaufmann, San Mateo, CA, 1989.

(Haussler, 1986) David Haussler. Quantifying inductive bias in concept learning. Technical Report UCSC-CRL-86-25, University of California, Santa Cruz, 1986.

(Haussler, 1987) David Haussler. Bias, version spaces and Valiant's learning framework. In *The Fourth International Workshop on Machine Learning*, pages 324–336, 95 First Street, Los Altos, CA 94022, 1987. Morgan Kaufmann Publishers, Inc.

(Hebb, 1949) Donald Hebb. *The organization of behavior.* Wiley (New York), 1949.

(Hetherington and Seidenberg, 1989) Phil A. Hetherington and Mark S. Seidenberg. Is there catatrophic interference in connectionist networks? In *Proceedings of the Eleventh Annual Cognitive Science Society Conference*, pages 26–33, Hillsdale, NJ, 1989. Lawrence Erlbaum Associates.

(Hinton and Sejnowski, 1986) G.E. Hinton and T.E. Sejnowski. Learning and relearning in boltzmann machines. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editor, *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations*, chapter 7, pages 282–317. MIT Press/Bradford Books., 1986.

(Hinton, 1987) G. Hinton. Learning translation invariant recognition in a massivelly parallel network. In *Proceedings of the European conference on parallel architectures and languages.* Springer, 1987.

(Hobson and Cheng, 1973) Arthur Hobson and Bin-Kang Cheng. A comparison of the Shannon and Kullback information measures. *Journal of Statistical Physics*, 7:301–310, 1973.

(Hofstadter, 1985) D.R. Hofstadter. Waking up from the boolean dream, or subcognition as computation. In *Metamagical schemas*, pages 631–665. New York: Basic Books, 1985.

(Holland *et al.*, 1986) John H. Holland, Keith J. Holyoak, Richard E. Nisbett, and Paul R. Thagard. *Induction: Processes of inference, learning and discovery.* MIT press, Cambridge, MA, 1986.

(Hopfield, 1982) J.J. Hopfield. Neural networks and physical systems with emergent collective computational properties. In *Proceedings of the National Academy of Science*, chapter 79, pages 2554–2558. U.S.A., 1982.

(Jordan and Jacob, 1990) M. I. Jordan and R. A. Jacob. Learning to control an unstable system using forward modeling. In D. S. Touretzky, editor, *Advances in neural information processing systems 2*, pages 324–331. Morgan Kaufmann, San Mateo, CA, 1990.

(Kohonen *et al.*, 1981) Teuvo Kohonen, Erkki Oja, and Pekka Lehtio. Storage and processing of information in distributed associative memory systems. In G.E. Hinton and J.A. Anderson, editors, *Parallel Models of Associative Memory*, chapter 4, pages 105–144. Lawrence Erlbaum Associates, 1981.

(Kohonen, 1977) T. Kohonen. *Associative memory: A system theoretical approach.* Springer-Verlag, New York, 1977.

(Kohonen, 1984) T. Kohonen. *Self-organization and associative memory.* Springer-Verlag, New York, 1984.

(Kolmogorov, 1965) A.N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–11, 1965.

(Kortge, 1990) Chris Kortge. Episodic memory on connectionist networks. In *The Twelth Conference of the Cognitive Science Society*, pages 764–771, Hillsdale, NJ, 1990. Lawrence Erlbaum Associates.

(Krogh and Hertz, To appear) Anders Krogh and John A. Hertz. Dynamics of generalization in linear perceptrons. *Obtained from the ftp internet Neuroprose database on archive.cis.ohio-state.edu*, To appear.

(Kukich, 1988) Karen Kukich. Back-propagation topologies for sequence generation. In *Proceedings of the IEEE Second International Conference on Neural Networks, San Diego*, pages 308–310. New York: IEEE, 1988.

(Kullback, 1959) S. Kullback. *Information theory and statitics.* John Wiley and Sons, Inc., New York, 1959.

(Le Cun *et al.*, 1990) Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Handwritten digit recognition with a back-propagation network. In D. S. Touretzky, editor, *Advances in neural information processing systems 2*, pages 396–404, San Mateo, 1990. Morgan Kauffman.

(Le Cun, 1985) Yann Le Cun. A learning procedure for asymmetric threshold networks. *Proc. Cognitiva*, 85:699–704, 1985.

(Legendre *et al.*, 1990a) Géraldine Legendre, Yoshiro Miyata, and Paul Smolensky. Harmonic grammar –a formal multi-level theory of linguistic well-formedness: Theoretical foundations. In *Proceedings of the Twelth Annual Cognitive Science Society Conference*, Hillsdale, NJ, 1990. Lawrence Erlbaum Associates.

(Legendre *et al.*, 1990b) Géraldine Legendre, Yoshiro Miyata, and Paul Smolensky. Harmonic grammar –a formal multi-level theory of linguistic well-formedness: An application. In *Proceedings of the Twelth Annual Cognitive Science Society Conference*, Hillsdale, NJ, 1990. Lawrence Erlbaum Associates.

(Legendre *et al.*, 1991) Géraldine Legendre, Yoshiro Miyata, and Paul Smolensky. Distributed recursive structure processing. In J. Moody R.P. Lippman and D. S. Touretzky, editors, *Advances in neural information processing systems 3*, pages 591–597. Morgan Kauffman, San Mateo, 1991.

(Lempel and Ziv, 1974) A. Lempel and J. Ziv. On the complexity of finite sequences. *IEEE Transactions on Information Theory*, IT-22:75–81, 1974.

(Leung-Yan Cheong and Cover, 1978) S.K. Leung-Yan Cheong and T.M. Cover. Some equivalences between Shannon entropy and Kolmogorov complexity. *IEEE Transactions on Information Theory*, IT-24:331–338, 1978.

(Marchand *et al.*, 1990) M. Marchand, M. Golea, and P. Rujan. A convergence theorem for sequential learning in two layer perceptrons. *Europhysics Letters*, 11:487–492, 1990.

(McClelland and Elman, 1986) J.L. McClelland and J.L. Elman. Interactive processes in speech perception: the TRACE model. In J. L. McClelland, D. E. Rumelhart, J. L., and the PDP Research Group, editor, *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 2: Psychological and Biological Processes*, chapter 15, pages 282–317. MIT Press/Bradford Books., 1986.

(McClelland and Rumelhart, 1986) J.L. McClelland and D.E. Rumelhart. A distributed model of human learning and memory. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editor, *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations*, chapter 17, pages 170–215. MIT Press/Bradford Books., 1986.

(McClelland and Rumelhart, 1988) J.L. McClelland and D.E. Rumelhart. *Explorations in Parallel Distributed Processing: A handbook of models, programs, and exercises.* MIT Press/Bradford Books, 1988.

(McClelland *et al.*, 1986) J.L. McClelland, D.E. Rumelhart, and the PDP group. *Explorations in Parallel Distributed Processing: Volume 2: Psychological and biological models.* MIT Press/Bradford Books, 1986.

(McCloskey and Cohen, 1989) M. McCloskey and N.J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In G.H. Bower, editor, *The Psychology of learning and motivation*, volume 23. New York: Academic Press, 1989.

(McCullogh and Pitts, 1943) W.S. McCullogh and W. Pitts. A logical calculus of the ideas immanent in the nervous activity. *Bulletin of mathematical biophysics*, 5:115–133, 1943.

(McDonald, 1985) Roderick P. McDonald. *Factor analysis and related methods.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1985.

(McMillan, 1991) Clayton McMillan. Personal Communication, Computer Science Department, University of Colorado, Boulder, 1991.

(Medin and Smith, 1984) D.L. Medin and E.E. Smith. Concepts and concept formation. *Annual review of psychology*, 35:113–138, 1984.

(Mezard and Nadal, 1989) M. Mezard and J.P. Nadal. Learning in feed-forward layered networks: The tiling construction algorithm. *Journal of Physics A*, 22:2191–2204, 1989.

(Michalski, 1983) R.S. Michalski. A theory and methodology of inductive learning. In *Machine Learning: An artificial intelligence approach*, pages 83–134. Tioga Press, 1983.

(Minsky and Papert, 1969) M. Minsky and S. Papert. *Perceptrons: An introduction to computational geometry.* The MIT Press, 1969.

(Mitchell, 1982) T.M. Mitchell. Generalization as search. *Artificial intelligence*, 18:203–226, 1982.

(Miyata, 1988) Yoshiro Miyata. Organization of action sequences in motor learning. In *Proceedings of the Tenth Annual Cognitive Science Society Conference*, Hillsdale, NJ, 1988. Cognitive Science Society, Lawrence Erlbaum Associates.

(Miyata, 1990) Yoshiro Miyata. *A user's guide to Planet version 5.6, a tool for constructing, running and looking into a PDP network.* Computer Science Department, University of Colorado, Boulder, December 1990.

(Mozer and Smolensky, 1989) Michael C. Mozer and Paul Smolensky. Using relevance to reduce network size automatically. *Connection Science*, 1:3–16, 1989.

(Mozer, 1990) Michael C. Mozer. *CONNEX: Graphical interface for connectionist networks.* Computer Science Department, University of Colorado, Boulder, 1990.

(Newell, 1980) A. Newell. Physical symbol systems. *Cognitive Science*, 4:135–183, 1980.

(Opper *et al.*, 1990) M. Opper, W. Kinzel, J. Kleinz, and R. Nehl. On the ability of the optimal perceptron to generalize. *Journal of Physics A*, 23:L581–L586, 1990.

(Otwell, 1990) Ken Otwell. Incremental back-propagation learning from novelty-based orthogonalization. In *Proceedings of the International Joint Conference on Neural Networks, Washington, D.C*, pages 561–564. New York: IEEE, 1990.

(Parker, 1985) David Parker. Learning logic. Technical Report TR-47, Center for Computational Research in Enconomics and Management Science, MIT, 1985.

(Pineda, 1987) F.J. Pineda. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 59:2229–2232, 1987.

(Pollack, 1988) Jordan Pollack. Recursive auto-associative memory: Devising compositional distributed representations. In *Proceedings of the Tenth Annual Cognitive Science Society Conference*, pages 33–39, Hillsdale, NJ, 1988. Cognitive Science Society, Lawrence Erlbaum Associates.

(Ratcliff, 1990) Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological Review*, 97:285–308, 1990.

(Rissanen, 1986) Jorma Rissanen. Stochastic complexity. *Annals of statistics*, 14:1080–1100, 1986.

(Rissanen, 1989) Jorma Rissanen. *Stochastic complexity in statistical enquiry.* World Scientific Publishing Company, Singapore, 1989.

(Rosenberg, 1987) Charles R. Rosenberg. Revealing the structure of NETtalk's internal representations. Technical report, Cognitive Science Laboratory, Princeton University, Princeton, New Jersey 08542, 1987.

(Rosenblatt, 1958) Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

(Rosenblatt, 1962) Frank Rosenblatt. *Principles of Neurodynamics.* Washington D.C.:Spartan Books, 1962.

(Ross, 1988) Sheldon Ross. *A first course in probability (third edition).* Macmillan Pubishing Company, New York, 1988.

(Rumelhart *et al.*, 1986a) D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editor, *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations*, chapter 8, pages 318–362. MIT Press/Bradford Books., 1986.

(Rumelhart *et al.*, 1986b) D.E. Rumelhart, J.L. McClelland, and the PDP group. *Explorations in Parallel Distributed Processing: Volume 1: Foundations.* MIT Press/Bradford Books, 1986.

(Rumelhart, 1988) David E. Rumelhart. Learning and generalization, 1988.

(Saito and Nakano, 1988) K. Saito and R. Nakano. Medical diagnostic expert system based on a pdp model. In *Proceedings of the IEEE Second International Conference on Neural Networks, San Diego*, pages 255–262. New York: IEEE, 1988.

(Samalam and Schwartz, 1989) V.K. Samalam and D.B. Schwartz. A study of learning and generalization by exhaustive analysis. Technical Report TM-0224-12-89-401, GTE laboratories, 1989.

(Sanger, 1989) Dennis Sanger. A technique for assigning local reponsibilities to hidden units in neural nets. Technical Report CU-CS-435-89, Department of Computer Science, University of Colorado at Boulder, 1989.

(Sanger, 1990) Dennis Sanger. *Contribution analysis: a technique for Assigning Local Responsabilities to Hidden Units in Neural Nets.* PhD thesis, University of Colorado, Boulder, 1990.

(Sankar and Mammone, 1991) Ananth Sankar and Richard F. Mammone. Optimal pruning of neural tree networks for improved generalization. In *Proceedings of the International Joint Conference on Neural Networks, Seattle, to appear.* New York: IEEE, 1991.

(Schwartz *et al.*, 1990) D.B. Schwartz, V.K. Samalam, S.A. Solla, and J.S. Denker. Exhaustive learning. *Neural Computation*, 2:374–385, 1990.

(Sejnowski and Rosenberg, 1987) T.J. Sejnowski and C.R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168, 1987.

(Servan-Schreiber *et al.*, 1989) D. Servan-Schreiber, A. Cleeremans, and J.L. McClelland. Encoding sequential structure in simple recurrent networks. In D. S. Touretzky, editor, *Advances in neural information processing systems 1*. Morgan Kaufmann, San Mateo, CA, 1989.

(Shannon, 1948) C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–422, 1948.

(Shannon, 1951) C. E. Shannon. Prediction and entropy of printed English. *The Bell System Technical Journal*, 30:50–64, 1951.

(Sirat and Nadal, 1990) J.-A. Sirat and J.-P Nadal. Neural trees: A new tool for classification. Technical Report Preprint, Laboratoire d'Electronique Phillips, Limeil-Brevannes, France, 1990.

(Sirat, 1990) J.-A Sirat. Personal Communication, 1990.

(Sloman and Rumelhart, To appear) S.A. Sloman and David E. Rumelhart. Reducing interference in distributed memories through episodic gating. To appear.

(Smolensky, 1983) Paul Smolensky. Schema selection and stochastic inferece in modular environment. In *Proceedings of the National Conference on Artificial Intelligence*, pages 33–39, 95 First Street, Los Altos, CA 94022, 1983. Morgan Kaufmann Publishers, Inc.

(Smolensky, 1987a) Paul Smolensky. The constituent structure of connectionit mental states: A reply to Fodor and Pylyshyn. *Southern Journal of Philosophy*, 1, Supplement:137–163, 1987.

(Smolensky, 1987b) Paul Smolensky. On variable binding and the representation of symbolic structures in connectionist systems. Technical Report CU-CS-355-87, Department of Computer Science, University of Colorado at Boulder, 1987.

(Smolensky, 1988) P. Smolensky. On the proper treatment of connectionism. *The behavioral and brain sciences*, March 1988.

(Smolensky, 1990) Paul Smolensky. Tensor product variable binding and the representations of symbolic structures in connectionist networks. *Artificial intelligence*, 46:159–216, 1990.

(Solomonoff, 1964) R.J. Solomonoff. A formal theory of inductive inference. *Information and Control*, 7:1–22, 1964.

(Solomonoff, 1978) R.J. Solomonoff. Complexity-based induction systems: Comparisons and convergence theorems. *IEEE Transactions on Information Theory*, IT-24:422–432, 1978.

(Sontag, 1990) Eduardo D. Sontag. Feedforward nets for interpolation and classification. 2:374–385, 1990.

(Stanfill and Waltz, 1986) Craig Stanfill and David Waltz. Towards memory-based reasoning. *Communications of the ACM*, 29:1213–1228, 1986.

(Staniland, 1966) A.C. Staniland. *Patterns of redundancy.* Cambridge at the University Press, 1966.

(Thagard and Nisbett, 1982) P. Thagard and R.E. Nisbett. Variability and confirmation. *Philosophical studies*, 42:379–394, 1982.

(Tishby *et al.*, 1989) N. Tishby, E. Levin, and S. Solla. Consistent inference of probabilities in layered networks: prediction and generalization. In *Proceedings of the International Joint Conference on Neural Networks, Washington, D.C*, pages 403–409. New York: IEEE, 1989.

(Valiant, 1984) L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

(Van Gelder, 1990) Tim Van Gelder. Compositionality: A connectionist variation on a classical theme. *Cognitive Science*, 14:355–384, 1990.

(Vapnik and Chervonenkis, 1971) V.N. Vapnik and A. Chervonenkis. On uniform convergence of relative frequencies of events to their probabilities. *Theory Prob. Appl.*, 16:264–280, 1971.

(Weigend *et al.*, 1990) Andreas S. Weigend, Bernardo A. Huberman, and David E. Rumelhart. Predicting the future: a connectionist approach. Technical Report Stanford-PDP-90-01, Stanford University, April 1990.

(Weigend *et al.*, 1991) Andreas S. Weigend, David E. Rumelhart, and Bernardo A. Huberman. Generalization by weight-elimination with application to forecasting. In J. Moody R.P. Lippman and D. S. Touretzky, editors, *Advances in neural information processing systems 3*, pages 875–882. Morgan Kauffman, San Mateo, 1991.

(Werbos, 1974) Paul Werbos. *Beyond regression: New tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard Universty, 1974.

(Willshaw, 1981) D. Willshaw. Holography, associative memory, and inductive generalization. In G.E. Hinton and J.A. Anderson, editors, *Parallel Models of Associative Memory*, chapter 3, pages 83–104. Lawrence Erlbaum Associates, 1981.

(Wolpert, 1990a) David Wolpert. Constructing a generalizer superior to nettalk via a mathematical theory of generalization. *Neural Networks*, 3:445–452, 1990.

(Wolpert, 1990b) David Wolpert. A mathematical theory of generalization, part I. *Complex Systems*, 4:151–200, 1990.

(Wolpert, 1990c) David Wolpert. A mathematical theory of generalization, part II. *Complex Systems*, 4:201–249, 1990.

(Young, 1971) F. Young, John. *Information theory.* London:Butterworth, 1971.

(Yu and Simmons, 1990) Yeong-Ho Yu and Robert F. Simmons. Incremental back-propagation learning from novelty-based orthogonalization. In *Proceedings of the International Joint Conference on Neural Networks, Washington, D.C.* New York: IEEE, 1990.

(Zemel *et al.*, 1990) R. S. Zemel, M. C. Mozer, and G. E. Hinton. Traffic: A model of object recognition based on transformations of feature instances. In D. S. Touretzky, editor, *Advances in neural information processing systems 2*, pages 266–273. Morgan Kaufmann, San Mateo, CA, 1990.

(Ziv and Lempel, 1978) J. Ziv and A. Lempel. Compression of individual sequences via variable-rate encoding. *IEEE Transactions on Information Theory*, IT-24:530–536, 1978.

(Zvonkin and Levin, 1970) A.K. Zvonkin and L.A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 25:83–124, 1970.