SECOND-ORDER GENERALIZATION
IN NEURAL NETWORKS

by

Ann Marie Grbavec
B.Sc.(Hons.) Queen's University 1990
B.Ed. Queen's University 1991
M.Sc. Queen's University 1995

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in the School
of
Computing Science

© Ann Marie Grbavec 2004
SIMON FRASER UNIVERSITY
Fall 2004

All rights reserved. This work may not be
reproduced in whole or in part, by photocopy
or other means, without the permission of the author.
Approval

Name: Ann Marie Grbavec
Degree: Doctor of Philosophy
Title of Thesis: Second-Order Generalization in Neural Networks

Examining Committee: Dr. Richard Vaughan
Chair

Senior Supervisor: Dr. Robert F. Hadley, Professor

Supervisor: Dr. Fred Popowich, Professor

SFU Examiner: Dr. Anoop Sarkar, Assistant Professor

External Examiner: Dr. Michael R. W. Dawson, Professor
Department of Psychology, University of Alberta

Date Approved: October 6, 2004
SIMON FRASER UNIVERSITY

PARTIAL COPYRIGHT LICENCE

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author’s written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

W. A. C. Bennett Library
Simon Fraser University
Burnaby, BC, Canada
Abstract

One of the main strengths of neural networks is their ability to generalize beyond training data. However, recent research suggests that certain types of generalization, which humans appear to perform readily, are problematic for traditional neural networks. This thesis examines the foundations of such claims and considers possibilities for resolving several issues they raise.

These forms of generalization have attracted attention in large part due to their implications about the role of symbol-processing in cognition. They have been shown to be beyond the scope of the types of neural networks that have been considered to offer an alternative to classically symbolic representations and rules: back-propagating multilayer perceptrons and simple recurrent networks. Interestingly, claims have been made that many classically symbolic machine learning techniques also fail to generalize in this way.

The tasks in question can be described, broadly, as generalizing relations to novel items. Previous formulations of the research problem have offered various specific but somewhat inconsistent criteria for characterizing this type of generalization. In this thesis, an analysis of previous formulations reveals how they are limited by unacknowledged assumptions about the role of representation, task form, and learning in problem equivalence. A framework for specifying a generalization task is introduced to support a more lucid discussion of these issues. Applying the framework to sample tasks reveals an underlying distinction between ways in which a generalization may be reached. Based on the results of the analysis, a more unified view of the problem space is presented and related sub-problems are identified.

Ways in which winner-take-all networks could play a role in the solution of two problem classes are outlined. Winner-take-all networks are considered to be more biologically plausible than the back-propagating networks which have been considered previously and unsuccessfully for the solution of such problems.
Acknowledgments

I am deeply grateful to my supervisors, Dr. Bob Hadley and Dr. Fred Popowich, for their support, advice, and encouragement throughout the course of this research. It has been an honour to work with them. I am especially indebted to my senior supervisor, Bob Hadley, for his very thoughtful feedback and for suggesting several of the possible counter-arguments to my own that are considered in this thesis. I would also like to thank the members of my examining committee for their humour and genuine curiosity. Finally, to the many kind friends, family, and strangers who have helped to make this work possible, my heartfelt gratitude.
Contents

Approval ii
Abstract iii
Acknowledgments iv
Table of Contents v
List of Tables viii
List of Figures ix
1 Introduction 1
2 Previous Approaches 5
   2.1 Universal Generalization 6
      2.1.1 Universally Quantified One-to-One Mappings 7
      2.1.2 Generalizing Outside the Training Space 7
      2.1.3 Eliminativist Models 9
      2.1.4 Training Independence 10
   2.2 Type-2 Generalization 11
      2.2.1 Inductive Justification 11
      2.2.2 Geometric Separability 13
   2.3 Related Problems 14
      2.3.1 Systematicity 15
      2.3.2 Cross-Category and Non-Category Generalization 16
      2.3.3 Relational Similarity 17
   2.4 Summary 19
3 Analysis

3.1 Generalization in Natural Cognition
  3.1.1 Relations
  3.1.2 Novelty
  3.1.3 Evidence for Type-2 Generalization

3.2 Issues in Evaluating Problem Equivalence
  3.2.1 Representation
  3.2.2 Task Form

3.3 Framework for Task Specification
  3.3.1 Generalization Task Specification
  3.3.2 Coarse and Ground Form Specifications

3.4 Summary

4 Reformulation and Classification

4.1 Common Ground
  4.1.1 Universal Generalization and Type-2 Generalization
  4.1.2 Related Problem Formulations

4.2 Underlying Distinctions
  4.2.1 Grounds for Generalization
  4.2.2 The Role of Learning

4.3 First-Order Generalization
  4.3.1 Standard First-Order Generalization
  4.3.2 Cross-Dimensional Generalization

4.4 Second-Order Generalization
  4.4.1 Core Relational Generalization
  4.4.2 Composite and Special-Purpose Relations

4.5 Summary

5 Techniques

5.1 Winner-Take-All Networks
  5.1.1 Techniques for Cross-Dimensional Generalization
  5.1.2 Training Independence
List of Tables

2.1 Universal Generalization Sample Problem [53, p. 37] ......................... 6
2.2 Type-2 Sample Problem [15] .................................................. 12

3.1 Data for Marcus’ Binary Identity Task ...................................... 32
3.2 Template for Generalization Task Data .................................... 48

4.1 Vector Similarity Task ............................................................ 65
4.2 Cross-Dimensional Generalization Task ....................................... 69
4.3 Sentence Completion Task ......................................................... 76
4.4 Similarity Matrix ................................................................. 77
4.5 Off-Diagonal Elements of the Similarity Matrix ......................... 78
4.6 Similarity Matrix for the Chair-Colour Task ............................... 79
4.7 Structure of Infant Grammar Task .......................................... 82
4.8 Structure of Relational Similarity Task ................................. 83
4.9 Linking Task ................................................................. 84
4.10 Classes of Generalization Tasks .............................................. 90

5.1 Comparable Cross-Dimensional Generalization Tasks .................. 95
5.2 Similarity Matrix ............................................................... 128
5.3 Similar Inputs ................................................................. 136
5.4 Dissimilar Inputs ............................................................... 137
List of Figures

2.1 Weight Update Rule for Back-Propagation ............................................. 10
2.2 Relational Similarity Task ................................................................. 18
3.1 Mapping of Data to Nodes of a Network ................................................. 49
4.1 Thornton's Relational Training Data [76, pp. 171-177] ............................ 68
4.2 Unlabelled Relational Training Data ...................................................... 68
4.3 Clustering of Training Inputs ............................................................... 70
4.4 Clustering Against Classification ......................................................... 71
4.5 Clustering of Training Cases ............................................................... 72
4.6 Subclusters Consistent with Class Labels .............................................. 72
4.7 Centroids of Local Clusters ................................................................. 73
4.8 Structure of the Sentence Completion Cases ........................................... 77
4.9 Internal Structure as Off-Diagonal Matrix Elements ............................... 78
4.10 Internal Structure of Phillips' Chair-Colour Task .................................. 79
4.11 Structure of Habituation Sentences ..................................................... 81
4.12 Internal Structure of a Task with Multiple Links ................................... 85
5.1 Architecture of Winner-Take-All Network ............................................. 92
5.2 WTA Network for Cross-Dimensional Generalization ............................... 96
5.3 Clustering of Training Inputs in the Original Input Space ....................... 97
5.4 Classification to be Learned ................................................................. 98
5.5 Expanded Inputs: Each Class Corresponds to a Separate Plane ............... 99
5.6 Maximum Cluster Size Within a Class Plane ......................................... 104
5.7 Centroids of Local Clusters ................................................................. 111
5.8 Clustering of Thornton’s Training Cases .............................................. 119
5.9 Relational Unit ..................................................................................... 125
5.10 Architecture of a Trace Module ........................................... 126
5.11 Portion of Relational Trace for Core Relational Generalization .......... 127
5.12 Architecture for Integrating Multiple Trace Modules ....................... 128
5.13 Connectivity of Prediction Vector ............................................. 129

A.1 Minimum Separation of Clusters in a Class Plane .......................... 155
A.2 Overlap of Clusters of Training Items ........................................ 155
Chapter 1

Introduction

Recently, researchers have identified certain generalization tasks which humans appear to perform easily, yet which present a profound challenge to traditional neural (connectionist) networks. By some accounts, even classically symbolic artificial intelligence techniques are challenged by this type of generalization. Although formulating the problem is the subject of considerable research and debate, it can be described, broadly, as that of generalizing relations to novel instances.

The problem has deep implications for Artificial Intelligence and Cognitive Science — not the least of which is the long-standing debate about the role of classical representations (i.e. symbols and rules) in intelligent systems. Works on the topic have provoked such response as to set records for volume of citations\(^1\). In addition to its theoretical value, progress in the area would have practical applications in many areas of Artificial Intelligence, such as scientific and mathematical discovery and natural language learning.

The types of generalization tasks in question have been characterized variously by researchers. But each formulation claims to identify a type of generalization that is so fundamental to intelligence that it presents a serious challenge to connectionism. This thesis examines the foundations of such claims and explores possibilities for resolving the problems they raise. Specifically, the thesis addresses the following questions:

\(^1\) Marcus reports that citation records were set by Elman's paper [19] and that his own work in the area [56] drew a response of at least nine different proposed models within a year of its publication [53, pp. 25, 59].
i) How meaningful are the previous formulations of the research problem?

ii) How do the formulations relate to each other?

iii) What underlies them?

iv) Can the formulations be improved or even unified?

v) How might unsupervised connectionist techniques help to address such challenges?

This thesis shows how previous problem formulations are limited by a number of unacknowledged assumptions and inconsistencies involving the role of representation, task form, and learning in evaluating problem equivalence. To clarify these, a framework for specifying a generalization task is proposed. Its application reveals distinct ways in which a generalization may be supported. Based on this, a more unified understanding of the underlying problem is reached. In particular, a distinction is drawn between First-Order and Second-Order Generalization. These terms describe two distinct and mutually exclusive classes of generalization tasks². Within each of these, further subclasses are identified. While most previous connectionist analyses are based on the results of back-propagation experiments, the more theoretically (and cognitively) plausible Winner-Take-All networks have remained relatively unexamined. This thesis proposes ways in which such networks could play a role in the types of generalization identified as problematic. Winner-Take-All networks are shown to address a subclass of First-Order generalization tasks — namely, Cross-Dimensional generalization — which contains tasks that have been considered to be problematic for back-propagating multilayer perceptrons. In addition, a novel technique (the Relational Trace method) is presented which uses Winner-Take-All networks to solve Core Relational generalization, an important subclass of Second-Order Generalization tasks.

The following chapter presents the research problem as it has appeared in the literature. The review outlines two main previous formulations of the research problem — Marcus’ Universal Generalization [52, 53] and Clark’s and Thornton’s Type-2 Generalization [15] — and

² Observe that First-Order generalization is thus neither a subclass, nor a superclass, of Second-Order generalization.
CHAPTER 1. INTRODUCTION

highlights related work: Cross-Category and Non-Category Generalization [63], Systematicity [24, 32, 60], and Relational Generalization [28].

The third chapter examines the foundations of the previous problem formulations. Arguments that motivate the identification of the main types of generalization as unique and significant are analyzed. Particular attention is paid to how problem equivalence is established and to how effectively the main class descriptions characterize sample problems. The analysis identifies and investigates a number of hidden assumptions, inconsistencies, and unresolved issues regarding representation and task form. A framework for specifying a generalization task is introduced to aid a clearer and more consistent treatment of these issues. The analysis addresses component i) of the thesis problem statement and establishes the foundation for exploring the remaining components in later chapters.

Chapter 4 explores ways in which previous problem formulations converge, overlap, or conflict. The newly proposed task specification framework helps to reveal an alternative understanding of the problem space. Issues concerning learning, similarity, and grounds for generalization are identified and explored as determining factors in a distinction between First-Order and Second-Order Generalization, proposed here, which may underlie disparate problem formulations. Based on these considerations, a more unified problem formulation is presented. Within each of the two main classes of generalization, subclasses are identified and their interrelations are elaborated. Standard First-Order generalization and Cross-Dimensional generalization are the subclasses of First-Order generalization. Second-Order generalization consists of the subclasses: Core Relational generalization and generalization involving Composite and Special Purpose Relations. Sample problems illustrate how these subclasses provide a coherent organization of the problem space. The unified formulation clarifies the main underlying research problem. It is this challenge which motivates the final phase of the research. The reformulation and classification address components ii), iii) and iv) of the thesis problem statement.

The fifth chapter describes how Winner-Take-All networks could address some of the challenges derived from the preceding analysis, reformulation, and classification. Winner-
CHAPTER 1. INTRODUCTION

Take-All networks are shown here to have formal properties that allow them to solve Cross-Dimensional generalization — a subclass of First-Order generalization problems that have been considered to be problematic for back-propagating multilayer perceptrons. This class contains certain types of tasks that would be classified as Type-2 Generalization tasks according to criteria presented by Thornton [73, 75].

The Relational Trace method is introduced and shown to address the central relational aspect of many Second-Order Generalization problems. An architecture for combining Winner-Take-All units to compute the Relational Trace is presented as a solution to Core Relational generalization — an important subclass of Second-Order Generalization problems. This class contains certain types of tasks previously described as Universal Generalization [52, 53] or as Cross-Category Generalization or Non-Category Generalization [63]. A proof-of-concept argument to support the plausibility of the approach is presented and its implications in the context of broader research issues are discussed. The techniques presented in this fifth chapter address question v), the final component of the thesis problem statement.

The final chapter presents the conclusions of the research, describes the main contributions of the thesis, and suggests possibilities for future work.

---

3 While the architecture would not be directly applicable to tasks that involve Composite or Special-Purpose relations (i.e. the final subclass of Second-Order Generalization), the discussion in Chapter 4 (Section 4.4.2) examines how the general method of computing the relational trace (using modules from the architecture) could be relevant in the solution of some such tasks.
Chapter 2

Previous Approaches

This chapter reviews how previous researchers have drawn distinctions between two types of generalization. Particular attention is paid to the ways in which the research problem has been formulated and to arguments about why a given formulation is considered to be significant and useful. Much previous work tends to be motivated by observations of apparent discrepancies between the generalization performance of human and artificial neural networks\(^1\). Some formulations present a challenge, not only for connectionism, but even for classically symbolic techniques [15, 63].

The researchers present various criteria for characterizing these challenging forms of generalization. More than one writer has observed that relations play a crucial role in such problems. Several views have been offered about how the more challenging generalizations extend to items that are novel in particular ways. In a few cases, reasons for the failure of the relevant artificial systems have been put forth.

Throughout this chapter, sample problems are used to illustrate the ideas presented. Each of the first two sections of this chapter is devoted to one of the main problem formulations: Marcus' *Universal Generalization* [52, 53] in Section 2.1; and *Type-2 Generalization* by Clark and Thornton [15] in Section 2.2. Related work is reviewed in Section 2.3: *Cross-Category and Non-Category Generalization* [63], *Systematicity* [24, 32, 60], and *Relational Generalization* [28]. The chapter concludes with a summary in Section 2.4.

---

\(^1\) For introductory material on neural networks, see [3, 42]. Supplementary context is available in [30].
2.1 Universal Generalization

In *The Algebraic Mind* [53], and earlier work² [52] Gary Marcus presents the following problem. Consider how you would complete the pattern shown in Table 2.1:

<table>
<thead>
<tr>
<th>Training Items</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Output</td>
</tr>
<tr>
<td>1010</td>
<td>1010</td>
</tr>
<tr>
<td>0100</td>
<td>0100</td>
</tr>
<tr>
<td>1110</td>
<td>1110</td>
</tr>
<tr>
<td>0000</td>
<td>0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Item</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>?</td>
</tr>
</tbody>
</table>

Marcus claims that the response of most humans is ‘1111’, whereas the types of neural networks that are most often explored as an alternative to classical (symbolic) computation would produce an output of ‘1110’. He reports this claim about human behaviour based on his own informal observation of individuals who were asked to imagine being trained on the input and output data in the table.

This example is considered to be a case of what Marcus calls *Universal Generalization*. Problems in this class are characterized as generalizing *Universally Quantified One-to-One Mappings* to novel items that lie *outside the training space*. Sections 2.1.1 and 2.1.2 outline these two key characteristics.

---

² The earlier work [52] presents a comparable but slightly larger data set than is shown here.
2.1.1 Universally Quantified One-to-One Mappings

The example in Table 2.1 is intended to illustrate the ability of humans to generalize what Marcus calls *Universally Quantified One-to-One Mappings (UQOTOM)*. A UQOTOM *maps each input to a single unique output* (and vice versa), since it is one-to-one. A mapping that is applicable to all instances in its domain is considered to be *Universally Quantified* [52, 53, p. 36].

In the example in Table 2.1, the mapping that is generalized is the *identity* function. Among the many possible examples of UQOTOM, Marcus emphasizes that identity and concatenation are particularly important. Concatenation, for example, is involved when forming the past or progressive tense of verbs or plurals of nouns. Other examples he identifies include reduplication (i.e. immediate repetition, used for pluralization in some languages), multiplication, and certain artificial grammars [53, pp. 36-45].

2.1.2 Generalizing Outside the Training Space

In the Binary Identity example (Table 2.1), generalization is considered to be occurring because a pattern in the training data (in this case, the identity relation) is extended to *novel* test items. In this example, the test item is novel with respect to the training items in two ways:

i) Neither the input (1111) nor the output (1111) has been presented before.

ii) The *values* of the right-most digit of the input and of the output are *novel* with respect to the training data. That is, the value of the right-most digit of the novel test input (and output) is ‘1’, whereas the corresponding positions of the training items never vary from a value of ‘0’.

It is this second type of novelty which is a key element of *Universal Generalization*. In Universal Generalization, test items are novel relative to the training items in that they fall *outside of the training space*. A *training space* is defined as “the area of the input space in which the training set is clustered” [53, p. 45].
CHAPTER 2. PREVIOUS APPROACHES

Suppose that each input or output of Table 2.1 is viewed as a point in a four-dimensional representation space, where its coordinates are the values of its digits — e.g. 1010 has coordinates (1,0,1,0). Then, any test input that has a value of ‘0’ in the right-most column is within the training space. Items with a value of ‘1’ in the right-most position are outside of the training space. In fact, this dimension is untrained in the sense that only one value (0) has ever been presented during training.

Marcus argues that testing for generalization outside the training space establishes whether the underlying UQOTOM (in this case identity) has been learned, rather than merely memorized as a finite set of input-output pairs. Two linguistic tasks involving universal generalization of identity are offered:

i) **Sentence Completion:** Given the examples: *a rose is a rose, a tulip is a tulip, a lily is a lily*, how would you complete the sentence: *a blicket is a ____?* Marcus claims, based on his own on informal observations, that humans can easily generalize from the examples to provide the answer blicket [52, 53, p. 50].

ii) **Infant Grammar:** Marcus claims that even infants can perform this type of generalization. In one experiment, seven-month-old infants are presented with spoken sentences from two artificial grammars [56]. The ABA grammar generates sentences like *ga na ga* and *li ti li*. The ABB grammar generates sentences such as *ga na na* and *li ti ti*. In the experiment, infants hear sentences from only one of the grammars for a habituation period of two minutes. Then, the experimenters observe how long the infants look at speakers playing sentences made up of entirely novel words that are either consistent or inconsistent with the habituation grammar. The results show that the infants tend to attend to the inconsistent items for a longer period of time than to

---

3 Each layer of a network (e.g. input, hidden, or output) has its own representational space, where each node corresponds to a dimension of the space. A representation space is sometimes also known as a feature space [42, p. 199], where each dimension of the space corresponds to a single feature.

4 For example, an item of the form ABB would be inconsistent if the habituation used the ABA grammar.
the consistent items. Marcus concludes from this that infants can generalize the grammatical rules (without, of course, necessarily being able to articulate those rules).

2.1.3 Eliminativist Models

Marcus claims that examples such as these illustrate a type of generalization that is central to intelligence and that, so far at least, has not been achieved by eliminativist neural networks. A network is called eliminativist if it does not manipulate classical symbolic representations; otherwise, it is considered to be implementational. Eliminativists challenge the symbol manipulation hypothesis, which claims that such symbolic representations are essential for cognition. But precisely what constitutes a symbol is subject to considerable debate. One interpretation of this is that a composite representation is classical if it cannot be activated without activating its (syntactic or semantic) constituents [24]. According to Marcus, the term 'symbol' is often use by researchers to mean merely a context-independent representation, which he considers to be quite ubiquitous [53, pp. 32-34]. So he focuses instead on symbol manipulation as the key factor, and elaborates on the symbol manipulation hypothesis. According to Marcus [53, pp. 3-4], there are actually three symbol manipulation hypotheses:

- "The mind represents abstract relationships between variables.
- The mind has a system of recursively structured representations.
- The mind distinguishes between mental representations of individuals and mental representations of kinds."

The types of networks that have been proposed as eliminativist models typically rely on back-propagation for training. Marcus [52] gives empirical evidence that back-propagating multilayer perceptrons are unable to generalize the identity function and that back-propagating simple recurrent networks (SRN's) [20] fail to generalize in the Sentence Completion example (a rose is a rose ...) to novel words that fall outside the training space.
2.1.4 Training Independence

The underlying reason for the failure of both of these types of networks is identified by Marcus as *training independence* (of both inputs and outputs) [52, 53]. That is, what one node learns is independent of what is learned by other nodes within the same layer. An examination of the weight update rule for back-propagation makes this property apparent.

Consider the back-propagation learning rule shown in Figure 2.1. If an input node \( i \) is never activated during training (\( x_i = 0 \)), then the weight change \( \Delta w_{ji} \) is zero, so nothing is learned about that node. If the input node is non-zero, but does not vary during training, that weight is determined independently of the activation values of any other input node, as can be seen from the weight update rule, and which Marcus confirms empirically [53, p. 47]. Thus, back-propagation exhibits *input independence*.

**Figure 2.1: Weight Update Rule for Back-Propagation**

\[
\Delta w_{ji} = \eta e x_i
\]

where \( \eta \) is the learning rate, and \( e = (t_j - x_j)x_i(1 - x_i) \) is the error rate.

Similarly, what one output node learns is independent of what any other output node learns. The weights feeding output \( j \) do not depend on the weights feeding any other output node, nor on the target values of any other node. So back-propagation learning also exhibits
CHAPTER 2. PREVIOUS APPROACHES

output independence. For similar reasons, unsupervised networks based on the Hebb rule, \( \Delta w_{ji} = \eta x_j x_i \) are also subject to training independence, and so will fail to generalize relations as Marcus specifies.

Since these learning rules are at the core of currently-known eliminativist models, Marcus concludes that there is still no evidence that relational learning is possible without some form of symbols, variables and rules. He invites researchers to explore how alternative connectionist models might address Universal Generalization.

2.2 Type-2 Generalization

Andy Clark and Chris Thornton [15] identify a class of generalization tasks which they claim cannot be solved by traditional connectionist techniques. Examples of such problems include parity, XOR, and the identity relation — when the test cases are not part of the training set. They claim that such problems are common in biologically realistic settings, including language learning, and give experimental data to show that back-propagation networks fail on sample problems of this class.

Clark and Thornton call this type of generalization task Type 2 or Relational — as opposed to Type 1 or Statistical tasks, which are tractable by standard neural and machine learning techniques. What distinguishes these two types of generalization, they claim, is the need for some systematic representational redescription (i.e. recoding — see Section 2.2.1) of the data, so that hidden statistical regularities become apparent.

2.2.1 Inductive Justification

Clark's and Thornton's analysis of the ways in which Type-2 Generalizations might be justified statistically gives a partial characterization of this class of problems. Type-1 Generalization tasks have solutions that can be justified in one of two ways. A given output value may be seen to be appropriate simply because it is the value that has occurred most frequently in the training cases. For example, consider the data shown in Table 2.2, where \( x_1 \) and \( x_2 \) are the inputs, and
$y$ is the output. For the test case, an output of $y = 1$ could be justified with a weak probability of $P(y = 1) = 0.6$, since 60 percent of the training cases have an output value of 1.

Table 2.2: Type-2 Sample Problem [15]

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
</tbody>
</table>

The more interesting form of Type-1 justification considers the conditional probabilities of the data. We may observe that a given output value $y$ is conditional upon some pattern of values appearing in the input. Suppose that the conditional probability $P(y \mid x_1 = a_1, x_2 = a_2, \ldots, x_n = a_n)$ is high among the training cases. That is, when one or more elements ($x_1, x_2, \ldots, x_n$) of the input vector $\vec{x}$ take on particular explicit values ($a_1, a_2, \ldots, a_n$), an output value of $y$ occurs with high probability. Then we have a justification for an output value of $y$ in response to test cases whose inputs also satisfy $(x_1 = a_1, x_2 = a_2, \ldots, x_n = a_n)$. For the example in Table 2.2, one possibility is $P(y = 1 \mid x_2 = 2) = 0.67$.

Clark and Thornton specify that Type-2 Generalizations are justified by a recoding of the data. Here, the input values play a less direct role in the conditional probabilities than they would in a Type-1 justification. In Type-2 generalizations, the output depends, not on the particular values of the inputs, but on a function of the inputs. For the example in Table 2.2, we might have observed that the values of $y$ vary with the difference between the two input variables. If we define a recoding function, $f(x_1, x_2) = |x_1 - x_2|$, then we have a strong indirect justification for an output of 0 for the test case, since $P(y = 0 \mid f(\vec{x}) = 0) = 1$. In general, Type-2
generalizations are justified by examining the conditional probability $P(y \mid f(\bar{x}) = \xi)$. Here, we find that a particular output value $y$ is likely when some recoding function $f$ of (some of the elements of) the input vector $\bar{x}$ has a particular value $\xi$.

Clark’s and Thornton’s representational redescription proposal includes the observation that the search space of possible recodings is infinitely large — equivalent to the space of all applicable Turing Machines. Representational redescription has been proposed as a learning and developmental hypothesis in Cognitive Science [14, 44]. The most complete proposal for applying the recoding strategy [5] outlines a neural framework, but leaves unanswered the essential question of how to find the appropriate recodings [30].

2.2.2 Geometric Separability

Another, somewhat inconsistent characterization given by Clark and Thornton is that Type-2 generalizations depend on the relative values of inputs, rather than on their absolute values in the representation space. In later work [73, 75], Thornton elaborates on this characterization and formalizes the view that the distinction between Type-1 and Type-2 problems is an extension (to higher dimensions) of Minsky’s and Papert’s distinction between linearly separable and linearly inseparable problems for perceptrons [59].

Thornton defines the Geometric Separability Index for a given task, $f$, as the proportion of nearest neighbour inputs which share the same output:

$$GSI(f) = \frac{\sum_{i=1}^{N} \{ [f(\bar{x}_i) + f(\bar{x}_i') + 1] \text{ mod } 2 \}}{N}$$

where $f$ is a binary target function (i.e. the output),

$\bar{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ is an input vector,

$\bar{x}_i' = (x_{i1}', x_{i2}', \ldots, x_{in}')$ is the nearest neighbour of $\bar{x}_i$,

and $N$ is the total number of input vectors in the data set.
So, for each pair of input vectors (say, \( \vec{a} \) and \( \vec{b} \)) that are nearest neighbours in the \( n \)-dimensional input space, if they also have the same binary output values (i.e. \( f(\vec{a}) = f(\vec{b}) \)), then the GSI is incremented. But if they have different output values, the GSI is not incremented. Thus, the GSI measures the degree to which proximity (similarity of the absolute values) of the input data predict similar output values.

Thornton’s Geometric Separability Index is a numeric value between zero and one which links the relationality of a problem to the degree of clustering of the data. It measures the degree to which inputs that are nearby in representation space are likely to belong to the same class. Problems where the proximity of inputs (similarity of absolute values) predicts similar output values have a high GSI value and are considered to be Type 1 problems. Type-2, or relational problems have a low GSI value. Thus, rather than being a Boolean predicate, as is Minsky’s and Papert’s notion, the GSI describes a continuum of separability.

Thornton uses the GSI to argue that even most symbolic machine learning algorithms are inadequate for Type 2 problems, since they tend to rely on a high degree of clustering of the data. The GSI of the most frequently used Machine Learning data sets, such as those in the UCI repository of Machine Learning Databases, have been shown to have high geometric separability. The GSI of sixteen of the most frequently used data sets ranges from 0.67 to 1.00, with an average value of 0.85. An interpretation of these results is that the machine learning problems most commonly studied are non-relational by design [73].

2.3 Related Problems

The main focus of the proposed research is on the two problem formulations reviewed above: Universal Generalization (Section 2.1) and Type-2 Generalization (Section 2.2). These two can be considered in the context of a number of related research problems. The relevant aspects of these are outlined in this Sections 2.3.2 to 2.3.3.

---

5 The GSI can be calculated using metrics appropriate to the data (e.g. Manhalobis metric for symbolic data or Euclidean for spatial data).
2.3.1 Systematicity

Although questions specifically concerning natural language learning are not the immediate focus of this thesis, an important generalization problem in that domain can be considered in relation to the research described above. First described by Fodor and Pylyshyn [24], systematicity, in its most general form, means that the ability to "entertain" certain thoughts necessarily entails the ability to entertain certain other systematically related thoughts. So in a sense, cognitive abilities are thought to 'come in clumps'. More specifically, linguistic systematicity (or systematicity of thought) concerns the capacity to understand systematically related sentences. For example, humans who can understand a sentence such as dogs chase cats can necessarily also understand the sentence cats chase dogs. More generally, the claim is that a sentence of the form $x \text{ Relation } y$ can only be understood by an intelligence that can also understand sentences of the form $y \text{ Relation } x$. Similarly, systematicity of logical inference suggests that certain inferences are systematically related to each other. For example, if one can conclude that "It is not raining", given the two facts: "If it is raining, then Mary will carry her umbrella", and "Mary is not carrying her umbrella", then one should be able to perform all other instances of modus tollens inferences.

Detailed hierarchies of types of systematic generalizations have been proposed by Hadley [32] and by Niklasson and van Gelder [60]. These specify how networks might generalize to test cases that have various kinds of novelty relative to the training cases. According to Hadley [31], weak systematicity involves generalizing to sentences containing novel combinations of words or symbols; strong systematicity requires generalizing to sentences that include words which appear in novel syntactic positions and at novel levels of embedding; and strong semantic systematicity [38] is strong systematicity with the ability to assign appropriate meanings to the novel test sentences. Niklasson and van Gelder [60] propose five levels of systematicity which are related, but not directly equivalent to Hadley's.

---

6 For a summary of the systematicity debate, see Hadley [36].
Systematicity is considered to be a key property of human intelligence, and so it has been presented as a challenge for connectionist systems. The extent to which eliminativist networks might satisfy the criteria continues to be a subject of research and debate. Controversial comparisons have been made between Universal Generalization and certain forms of Systematicity (see Section 4.1.2). But the problem of training independence appears to have relevance to both forms of generalization. To the extent that training independence may limit the abilities of networks to acquire cognitive abilities that 'clump' as do human abilities, it is intimately related to the general problem of systematicity. In particular, training independence has been shown to be an obstacle for strong systematicity in back-propagating perceptrons and simple recurrent networks when local\(^7\) input/output representations are used [62].

### 2.3.2 Cross-Category and Non-Category Generalization

Motivated by the failure of back-propagating perceptrons and simple recurrent networks to support strong systematicity (see Section 2.3.1) with local input/output representations, Phillips [63] considers whether appropriate distributed representations could resolve the dilemma. He argues that the availability of the relevant distributed representations cannot be assumed. Phillips describes types of generalization that he says humans perform without access to the kinds of non-local representations that perceptrons or SRN's would need.

In Cross-Category Generalization, generalizations are made to novel items that do not share a common class with training items and are unlikely to have been encountered in similar common contexts. Thus, it is very unlikely that similar (i.e. nearby in representation space) internal representations are available.

Phillips gives an example of this type of generalization: Suppose subjects have learned to respond to the question: “John painted the chair red. What colour is the chair?” with the answer red. Then, claims Phillips, they will be able to respond appropriately (generalize) to

---

\(^7\) By local, Phillips means that only one node is activated at a time. For example, the vectors 1000, 0100, 0010, 0001, could represent four distinct inputs.
questions of this form which involve regularly novel items, such as green and blue, even though they have likely not encountered these specific inferences before.

The regularly novel items belong to the same category (i.e., colour) as the training item, red. But Phillips claims that humans can also generalize to radically novel items (i.e., items taken from other categories), such as mango, which is unlikely to have been learned as a type of colour. Thus, mango is represented on different units than are the regular colours, which does not provide sufficient similarity for generalization by back-propagation.

Further, Phillips claims that unilingual English speakers can also generalize to such radically novel items as aka (Japanese for red), which almost certainly has not been learned as an instance of colour, or of any other category. In this type of generalization (non-category generalization), it is most unlikely that an internal representation aka that is similar to that for red could have been learned prior to the generalization task.

Since standard, first-order connectionism (i.e., back-propagating perceptrons and SRN’s) depend on similarities that do not seem to be present in this type of generalization, Phillips concludes that the models are not adequate to explain such generalizations. Instead, he proposes exploring alternative models with symbol-like capacities, such as tensor networks. These perform binding by calculating the outer product of a vector representing data and a vector which explicitly represents relations [69, 64].

### 2.3.3 Relational Similarity

Much of the previous research presents generalization challenges that are motivated by human performance. However, claims about human generalization are commonly made without clear supporting arguments or experimental evidence. The experiment outlined below sheds some light on the types of generalization that human subjects can perform.

---

8 Phillips acknowledges that current context might be useful for constructing usefully similar internal representations, as outlined in Boden’s and Niklasson’s context similarity proposal [9]. However these would not be available prior to the generalization task, and so are not specifically relevant to his argument.
Consider the diagram in Figure 2.2. Which of the lower pairs of objects is most similar to the top pair?

![Diagram of a relational similarity task](image)

**Figure 2.2: Relational Similarity Task**

When this task was presented to subjects in a psychology experiment [28], the pair on the left was frequently chosen by children. However, adults chose the pair on the right 70% of the time. The pair on the left appears most similar *physically* to the top pair, whereas the pair on the right is most *relationally* similar to the top pair. To explore this distinction further, the experimenters examined the effects of bias. When a bias was introduced in the instructions in order to encourage looking for relationships, adults chose the pair on the right 90% of the time. When instructions were biased toward seeking matching attributes, they chose the same pair only 20% of the time.

The experiment supports intuitions that adults often base similarity judgements on such high-level relations, rather than more basic physical or perceptual commonalities. Results such as these highlight the flexibility of human judgements of similarity and are compatible with the possibility that relational judgements might be a relatively complex or late-developing ability in humans.


2.4 Summary

In this chapter, a review of the literature shows how previous researchers have identified particular types of generalization that are problematic for many connectionist networks. Marcus [52, 53, p. 47] characterizes Universal Generalization tasks as those which generalize Universally Quantified One-to-One Mappings (UQOTOM) to novel items that fall outside the training space. An item falls beyond the training space when it presents a novel activation value to at least one node that has not varied during training. Marcus claims that humans readily perform such generalizations, but traditional eliminativist neural networks do not, due to the problem of training independence. That is, what one node learns is independent of what is learned by other nodes within the same layer.

Clark and Thornton identify a class of generalization tasks — Type-2 Generalization — that appear to be regularly solved by biological learners, but which are beyond the scope of many connectionist and even symbolic learners [15]. Thornton characterizes such Type-2 Generalization tasks as those which have low relevant geometric separability [73, 75]. The Geometric Separability Index (GSI) is a value between zero and one which measures the degree to which inputs that are nearby in representation space are likely to belong to the same class. By contrast, Type-1 problems, which are more readily handled by traditional connectionist and symbolic learners, would have a relatively high GSI. Clark and Thornton purport that the solution to Type-2 problems depends on finding a representational redescription (i.e. a recoding function) of the data, such that there is a correlation between the values of the training outputs and the recoding of their corresponding inputs [15].

Related problems reviewed in this chapter include Systematicity, Cross-Category Generalization, Non-Category Generalization, and Relational Similarity. Systematicity is the ability to process new thoughts, sentences, or inferences that are systematically related to those which have been previously encountered. Multiple formulations of the problem exist (e.g. [24, 32, 60]), which vary notably in the type and degree of novelty present in the training cases. Some such tasks challenge not only eliminativist networks, but even classically symbolic methods [38, 12].
Cross-Category and Non-Category generalization extend to novel items that are so radically novel (i.e. dissimilar from those in the sample cases) that eliminativist networks are unable to recognize that they should be handled similarly to those in the sample cases [63]. It has been suggested that this limitation may again be due to training independence. Finally, Relational Similarity has been identified in studies of human subjects as one of at least two types of similarity judgements people can make [28].

Because Universal and Type-2 Generalization provide the most explicit formulations in terms of representation and task form, they are the main focus of the analysis in the following chapter. The related problems are revisited in Chapter 4.
Chapter 3

Analysis

In this chapter, an analysis of previous problem formulations reveals important inconsistencies and unstated assumptions about task equivalence. These are addressed by identifying and clarifying specifically relevant issues involving relations, novelty, representation, and task form. Together, these considerations provide a foundation for the more unified problem formulation and classification outlined in Chapter 4.

The analysis addresses component i) of the thesis research problem described in Chapter 1 and establishes the foundation for the study of the remaining components in ensuing chapters:

i) How meaningful are the problem formulations?

Section 3.1 explores the relationship between the problem formulations and human generalization capabilities. In doing so, it raises fundamental questions about representation and task form in evaluating problem equivalence, which are discussed further in Section 3.2. To help address some of these issues, a framework for specification of generalization tasks is introduced in Section 3.3. The chapter concludes with a summary in Section 3.4.

3.1 Generalization in Natural Cognition

One of the main reasons for distinguishing the types of generalization problems researchers have been discussing is that they seem to reveal a wide gap between human and machine capabilities. The authors of both main problem formulations — Marcus’ Universal Generalization [52, 53] and Clark’s and Thornton’s Type-2 Generalization [15] — claim that
humans readily perform the types of generalization they describe. These claims raise important issues about representation and task form in evaluating task equivalence. The most directly relevant formal studies of human subjects are presented by Marcus. For this reason, Universal Generalization is explored first in some depth to show how these issues arise. Clark’s and Thornton’s claims are examined in Section 3.1.3.

Central to Marcus’ problem formulation of Universal Generalization (Section 2.1) is the challenge of generalizing universally quantified one-to-one mappings (UQOTOM) to novel test cases that lie outside of the training space. He claims that humans can and ‘routinely’ do perform exactly this type of generalization [52, 53, pp. 36-39, 46, 50]. Along with a number of sample problems available for informal and introspective examination, two formal experimental studies have been put forward as empirical evidence for this claim about human generalization capacities: a study of Infants’ habituation to artificial grammars by Marcus et al. [56] and a study of Hebrew word formation by Berent et al. [8].

This section considers whether the behaviours observed, particularly in the formal studies, actually satisfy the criteria put forward in Marcus’ own problem formulation. That is, does Marcus’ formulation accurately describe the observed generalization capacities of subjects? This would require establishing both of the following two points:

i) that the relations being generalized by the subjects are, in fact, universally quantified one-to-one mappings (UQOTOM) between inputs and outputs, and

ii) that the generalization is made to novel items that fall outside the training space.

These two criteria are examined in Sections 3.1.1 and 3.1.2 with respect to both the formal and informal forms of evidence Marcus provides.

3.1.1 Relations

Marcus does not intend the first part of his problem formulation — the criterion for UQOTOM — to apply only to artificial networks. Rather, the claim that humans can and commonly do generalize UQOTOM is essential to his larger argument, as he clearly indicates
CHAPTER 3. ANALYSIS

[52, 53, pp. 36-39, 50]. This claim is based on two forms of evidence: informal observation, including introspection, and formal experimental studies.

A Universally Quantified One-to-One Mapping (UQOTOM) is a function that maps each input of its domain to a unique output [53, p. 193]. Marcus is very explicit that “In universally quantified one-to-one functions, it does not suffice to assimilate each new response to an already known category, because each new input maps to a new output. The right answer for a new input in the functions will thus not be a category that the network has seen before” [52]. This criterion appears to be satisfied by the informal examples in a way that is less apparent in the more formal studies to be discussed below.

Evidence from Informal Observation and Introspection

One way in which human behaviour might be described in terms of the inputs and outputs of a UQOTOM is suggested by two sample problems: the Binary Identity task1 and the Sentence Completion task2. Marcus presents the Binary Identity task as an example of a Universal Generalization problem which humans can solve [53, p. 37]3. As discussed in Chapter 2 (Section 2.1), he reports that in his informal observation, when a person is given the instruction “imagine that you are trained on the input and output data given”, he or she will respond to the novel input with “1111” [53, p. 50]. Similarly, although without reporting all of the training data or describing evidence for this belief, Marcus states that humans can generalize the Sentence Completion task to the novel word blicket [29, 53, p. 50]. In both of these cases, subjects are considered to be generalizing UQOTOM to novel test cases.

In these two informal examples, subjects show that they can generalize this UQOTOM to novel cases by responding with the specific unique predicted value. In the Binary Identity task, the correct response to a novel input of ‘1111’ is the novel output ‘1111’. In the Sentence

---

1 See Table 3.1 on page 32 and Section 2.1 for details of the Binary Identity task.
2 In the Sentence Completion task (Section 2.1.2), subjects are given the sentences: a rose is a rose, a tulip is a tulip, a lily is a lily, then asked to complete the sentence: a blicket is a ?
3 Somewhat different training data are presented in an earlier paper [52], but the differences are irrelevant to his argument.
Completion task, the appropriate response to a novel input of *A blicket is a ___* is the novel output *blicket*. In both of these cases, a person's observable behaviour — a spoken or written response — is readily interpreted as a specific unique response to the test input. This is consistent with the characterization of the tasks as involving a UQOTOM between inputs and outputs. Thus, these two examples seem to suggest — if informal observation and introspection can be trusted — that humans are able to generalize universally quantified one-to-one mappings between inputs and outputs.

**Formal Studies of Human Subjects**

However, in the two formal studies designed to focus on Universal Generalization in humans, the UQOTOM requirement for a specific unique response for each input is not clearly satisfied in the same way as in the less formal examples discussed above. What is directly observed is not a specific unique response related by a UQOTOM to each input. Rather, subjects' observable responses fall into a small number of categories (2-5).

Recall that in the infant grammar study [55] (see section 2.1.2), infants were habituated to sentences from one of two grammars — either of the form ABA (e.g. *ga na ga*) or ABB (e.g. *ga na na*). Then, test sentences were presented to the infants from a grammar that was either the same as (i.e. consistent) or different from (i.e. inconsistent) the grammar of the habituation sentences. The observed response was the length of time that an infant attended to the speaker. The experimenters found a statistically significant difference between the mean length of time for which infants attend to consistent vs. inconsistent test items. But such an observation can only distinguish two types of responses to two classes of inputs (i.e. either consistent with habituation or inconsistent). The behaviour that is directly observed — the actual length of time an infant attends — is not related in a one-to-one way with the input. It is, rather, a many-to-one mapping, where each item is mapped to one of two separate ranges of responses. Thus, what is directly observed is a discrimination task (or perhaps an attention task), not one where

---

4 The question of whether they are also generalizing outside of the training space, thereby satisfying the second element of Marcus' criteria for Universal Generalization, is discussed in Section 3.1.2.
each input corresponds to a unique response. Even if we consider the grouping of outputs into two classes to be a *post hoc* interpretation, it remains that the observed response itself (i.e. the length of time an infant attends) cannot reasonably be interpreted as being systematically related to the inputs (i.e. the data presented to the subjects) by a UQOTOM.

Although Marcus does not address this inconsistency, one possibility for reconciling the observations with his definition of Universal Generalization is considered here. In order to interpret an infant’s behaviour as satisfying the UQOTOM criteria of Universal Generalization, an assumption might be made that the infant is making a specific internal prediction about what the final word is expected to be. That is, one might argue that the observed behaviour could be explained by an internal process that satisfies the UQOTOM criteria of Universal Generalization.

For the first two experiments of the infant grammar study, it is possible, although not necessary, for the infants to generate a specific internal prediction for the final word. For example, habituation on sentences of the form ABB, followed by the first two words of a novel test sentence (such as *li ti*), could allow subjects to form an expectation about the specific value of the third word (in this case, *ti* would be consistent with habituation). Encountering a discrepancy between the actual value of the final word and its expected value might then lead the infants to attend longer. Alternatively, the infants might simply be forming an expectation about whether the third word is the same or different from the preceding word(s). The observed behaviour could equally well be a response to the violation of this simpler type of expectation. So the stronger assumption (i.e. that the task involves predicting a specific value) is by no means necessary to explain the experimental observations.

Marcus et al. [56] observe that in the first two experiments of their study, infants might simply be responding to an even more basic property which distinguishes the two grammars: the presence or absence of immediate reduplication in the training and test sentences [56]. Citing this reason, they include in their study a third experiment which uses the same protocol

---

5 Likewise, for habituation on sentences of the form ABA, a specific prediction for the third word could be generated.
as the first two experiments, but in which both grammars contain immediate reduplication (i.e. AAB vs. ABB).

In the third experiment, habituation on the ABB grammar could allow subjects to generate a prediction about the specific value of the final word. But such a prediction is not necessary in order to account for the observed behaviour, since, for example, an expectation that the second word should differ from the first would be sufficient to discriminate between the two grammars. Further, habituation on sentences of the form AAB gives no way to predict a specific value for the final word, since it is always different from each of the first two words during habituation. If, for this particular case, the ‘output’ is taken to be the second word (only), then a specific prediction would be possible. But again, such a prediction is not necessary, since simply predicting that the third word will be different from the second would be sufficient to distinguish the grammars. Finally, the authors of the study themselves observe that the tasks in the third experiment could have been performed by infants who paid attention to only the two final words of each sentence [56]. Under the AAB habituation condition, ignoring the first word means that the ‘output’ would be final word, which cannot be predicted.

So in none of the three experiments of the study is it necessary to assume that the infants are generating an internal prediction of the specific value of the ‘output’. Although the infants do appear to be generalizing in an interesting and perhaps important sense, it is not one which clearly satisfies Marcus’ definition of Universal Generalization. The attention responses observed in all three experiments of the infant study might be adequately explained by a capacity of infants to discriminate UQOTOM between inputs. Perhaps the infants can even generalize this discrimination to novel inputs that are outside the training space (see Section 3.1.2). But to conclude that they must be generating a specific, unique output (prediction) related to each input by a UQOTOM is questionable, since such predictions are neither the only nor the simplest way to explain the observed behaviour. Until there is reason or evidence that such an assumption is likely to be the case, the question of whether the tasks in the study satisfy Marcus’ UQOTOM criteria remains open to doubt.
This difference between two task forms is critical to Marcus' formulation of the problem. Universal Generalization requires generating a unique novel output for each input, according to some UQOTOM that holds between inputs and outputs. Discriminating (or classifying) inputs into a small number of classes is not a case of Universal Generalization, as Marcus and his colleagues emphasize [8], but then seem to overlook, in their study of Hebrew grammatical generalization. In this more recent collaborative work with Berent et al. [8], Marcus presents further experimental evidence intended to show that humans — in this case adults — can generalize UQOTOM. Unlike the infant grammar experiments, studies of adults need not rely on interpretations of a limited range of behavioural responses (e.g. duration of attention). Unfortunately, the design of this recent study does not take advantage of the potential abilities of adults to respond with a behaviour that is unique for each input. Instead, each of these experiments is designed so that subjects need only distinguish between classes of inputs. Each experiment only distinguishes between 2 to 5 different types of responses. The experiments investigate how a particular constraint on Hebrew word formation can be generalized to novel items. The constraint disallows words with a root structure of the form XXY (i.e. where the first two syllables of the word root begin with identical consonants). In experiment 1, subjects are asked to rate each test word (1 = best, 2 = intermediate, and 3 = worst) depending on the degree to which a test word "sounded like a possible Hebrew word" [8]. In experiment 2, a rating scale of 1-5 (1 = impossible; 5 = excellent) is used. In experiment 3, subjects provide timed yes/no responses using a keyboard to indicate whether or not a word corresponds to an existing Hebrew word.

Subjects are considered to be generalizing a UQOTOM (in this case, the prohibition on the XXY pattern of repetition in word structure). This is clearly a UQOTOM (in this case, the identity relation), but it is between syllables that are presented to the subjects, not between inputs and outputs. There is no observation that subjects are producing a unique novel response for each input (not to mention, a response that is systematically determined by a particular input-output relation). Subjects are never asked, for example, to respond with the second syllable, based on the first. Rather, they are asked to determine whether (or to what degree)
input syllables stand in this relation to each other. Even if the similarity ratings or response times in the Hebrew word formation study are treated as continuous variables, rather than as a group of categories, the observed responses do not correspond by one-to-one mappings with the inputs. As in the infant study, the mapping does not determine a unique behaviour for each input.

A possibility for reconciling the observations with the requirement for generation is to assume that the relevant unique response is being generated internally. That is, the subjects in the Hebrew word formation study could be assumed to be internally generating a specific (unique, novel) prediction about the second syllable, related by a UQOTOM to the first. The need for this unstated, hence unsupported, assumption is somewhat surprising, given the pains that are taken by the authors to emphasize that generation, not classification, is what is required for models (connectionist or otherwise) to satisfy the Universal Generalization challenge: “The systems studied by Eddington, however, only classify an item into two predetermined categories (regular vs. irregular) — they do not generate the past tense form. This seemingly minor procedural modification is significant, since it overrides the problem of copying the stem (Marcus 2001)” [8]. But Berent et al. do not make it clear why the tasks in their study are not also a form of classification, as they appear to be at the level that is directly observable. Any assumption that specific predicted values are being generated internally would seem to require some additional support, before the Hebrew language study should be accepted as a clear case of Universal Generalization in humans subjects.

Given that the infants in the habituation experiments appear to have performed tasks of a comparable form, where generating a specific internal prediction was unnecessary, it is conceivable that the adults in this study might also not need to do so. The infants need only have generated a prediction about whether the next syllable should be the same or different from the last; attention would be aroused when the expectation is violated. Subjects in the Hebrew word formation study must determine, only after hearing or reading a word, whether the first two syllables of the word root begin with identical consonants. Why such a judgement requires generating, even internally, the specific consonant value that is disallowed in the second
syllable is far from clear. Without some support for such an assumption about internal generation, it appears that Berent et al. are testing only the abilities of subjects to classify (i.e. distinguish between classes of) word formation, rather than to actually generate a unique response that is related to each input by a common UQOTOM.

Implications

If informal observations and introspection are trustworthy, there might seem to be nothing in principal preventing direct empirical evidence that humans — at least adults — can generalize universally quantified one-to-one mappings between inputs and outputs. But so far, the strongest evidence Marcus reports is informal, introspective, or indirect. In his formal studies, humans were observed to discriminate or classify UQOTOM between inputs (i.e. between data items presented as part of the training). However, in the definition of Universal Generalization, particularly when evaluating models, Marcus requires generation of a unique novel output (i.e. the response produced by the subject or network) related by a UQOTOM to each input.

This inconsistency has important implications for Marcus’ larger purpose concerning the role of symbol manipulation in cognition. He argues that, since humans do appear to perform Universal Generalization, but current eliminativist models cannot, symbol manipulation seems more likely to provide an adequate explanation [52, 53]. But, as this section has revealed, Marcus’ most formal evidence for his premise about Universal Generalization in humans does not clearly satisfy his own definition of Universal Generalization. He shows that eliminativist models of Universal Generalization cannot perform generation tasks, whereas the subjects in his formal studies need only perform discrimination or classification tasks. Thus, Marcus’ larger argument is weakened by the inconsistencies in his formulation of the problem.

In order to strengthen his argument, Marcus could adopt a modified stance where the definition of Universal Generalization includes, not only generating the unique novel outputs, but also classifying UQOTOM between inputs (i.e. between data items presented as part of the training). He would then need to show that the eliminativist networks in question also cannot
perform tasks of this form\(^6\). Alternatively, Marcus could attempt to justify why models of Universal Generalization must generate a unique novel response, when discrimination or classification tasks are considered sufficient as evidence of Universal Generalization in humans.

While the limitations of Marcus’ problem formulation have implications for his larger argument about symbol manipulation, they do not mean that all of his claims are incorrect. They do not undermine his claim that feedforward multilayer perceptrons and simple recurrent networks cannot perform Universal Generalization as he defines it. Nor do they disprove his conclusion that cognition likely includes symbol manipulation. His conclusion may be true or not, but his argument gives less support to it than Marcus suggests. The validity of his argument is weakened by the difficulties, described in this and the following section, in establishing his claim that humans perform Universal Generalization. If it has not been clearly established that humans perform Universal Generalization, then the significance of the failure of eliminativist methods to do so, and the meaningfulness of his problem formulation would also be less than they might initially appear. The following section examines the evidence that humans can satisfy the second criteria of Universal Generalization — generalizing beyond the training space.

### 3.1.2 Novelty

In addition to the requirement involving UQOTOM, Universal Generalization also requires that test items are novel in the sense that they fall outside of the training space. Marcus clearly intends the novelty (i.e. training space) component of his formulation to be descriptive of human generalization abilities. He cites it as motivation for one of his empirical studies:

\(^6\) This is not to argue that such classification tasks could be accomplished by back-propagating networks. Indeed, it seems unlikely that they could be. Rather, the point being made here is that Marcus’ definition of Universal Generalization provides a formulation of the problem that is less useful in supporting his argument than it appears at first glance. Under either the original or ‘modified’ definition of Universal Generalization, Marcus’ argument is, so far, incomplete, since equivalence between the tasks performed by networks and human subjects has not been clearly established.
"Although the "within" vs. "outside" of training space distinction gives us an important yardstick for evaluating models, that yardstick must be compared with human behavior" [8]. But applying this criterion to human generalization is somewhat problematic, since a training space is defined in terms of how data are mapped to the nodes of a network — factors we cannot directly observe in humans. That is, without knowing how the data of a particular task are mapped to the nodes of a network, the training space is undefined, since the dimensions of the representational space are unknown.

Claims that humans are generalizing outside the training space depend on how data are mapped to nodes. Test items that provide a new value to at least one node which has not varied during training are considered to fall outside of the training space. So establishing that humans are in fact generalizing outside the training space would require knowing that their internal representation of the novel item includes a novel value for at least one node which did not vary during training. Until neural imaging techniques improve, even the most formal empirical studies designed to show Universal Generalization in humans must rely on assumptions or careful arguments about internal representation.

Geometry of a Training Space

Before discussing the evidence and arguments, some ambiguities in Marcus' presentation of the geometry of a training space require clarification. Marcus specifies that the training space is a "subspace" of the input space, "delimited by the values of the features that appeared in the training set"[52]. This might be taken to mean that the training space is a subspace in the algebraic sense of the term — which contains the training vectors. For example, if the training set consists of the two vectors [1 0 1] and [0 1 1], then the training space might seem to be the two-dimensional subspace (of the three-dimensional input space) that is spanned by those two vectors. By definition of a subspace, this would be a plane that passes through the origin,

7 Representation space and input space are introduced in Section 2.1.2.
8 The relevant definitions are available in most linear algebra textbooks (e.g. [72, pp. 63 - 65, 83]).
as well as through the two points whose coordinates are $(1, 0, 1)$ and $(0, 1, 1)$. This plane would consist of exactly those vectors that are linear combinations of the two training vectors.

But this does not seem to be the meaning that Marcus intends. He states that, if a training set consists of the two vectors $[1 \ 0 \ 1]$ and $[0 \ 1 \ 1]$, then the test item $[1 \ 1 \ 1]$ would be within the training space, even though, as he points out, it is not a linear combination of the training items [52]. So, instead of a subspace, what Marcus seems to have in mind here as a training space is a plane whose normal is the z-axis and which intersects the z-axis at the point $(0, 0, 1)$. Since this plane does not pass through the origin, it is not a subspace of the input space. It is this latter conception of a training space (i.e. not as an algebraic subspace) that is most consistent with Marcus' use of the term and which is used in the remainder of this thesis.

Evidence from Informal Observation and Introspection

Two examples of Universal Generalization tasks are provided as informal evidence that humans can generalize beyond the training space: the Binary Identity task (see Table 3.1 on page 32 and Section 2.1) and the Sentence Completion task (Section 2.1.2) [52, 53]. Table 3.1 shows the data used in the Binary identity task.

**Table 3.1: Data for Marcus' Binary Identity Task**

<table>
<thead>
<tr>
<th>1 0 1 0</th>
<th>1 0 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 0</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>1 1 1 0</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>?</td>
</tr>
</tbody>
</table>

---

9 Nor is it a "vector space", as might be understandably concluded [35] — at least not in the algebraic sense of the term — since all vector spaces, by definition, include the origin.
Here, the right-most digit of each output (and of each input) never varies from zero during training. Since the novel test item “1 1 1 1” contains a value of 1 in those positions, this test item falls outside the training space.

Marcus compares how humans and artificial neural networks might perform this task. But his discussion depends on the assumption that each column corresponds to a separate node of the network. In order for the Binary Identity task to provide support for the claim that humans are generalizing outside the training space in the way that Marcus describes, we would have to be convinced that, in humans, the novel value of “1” in the representation of the right-most column includes at least one node that has not varied during the training. In what way does an adult’s perception or conception of the written symbol ‘0’ correspond to a single inactive node which takes on a non-zero value only in response to the test case? Marcus presents no arguments to suggest why such an internal representational scheme would be the case in humans who are performing this task. In order for this informal example to be more than illustrative, Marcus would need to address the question of whether this or other internal representations\(^ {10} \) that satisfy the training space criteria are likely to be the case in humans. So, although the Binary Identity example might conform to the relational (i.e. UQOTOM) aspect of Universal Generalization, the training space requirement seems less clearly satisfied than the relational one, since it is defined in terms of unobservable and unsupported details about how humans internally represent the data.

Similarly, claims that the Sentence Completion task (Section 2.1.2) is a case of human generalization beyond the training space would require consideration of how the data are represented internally by humans. In this case, Marcus does provide some discussion of his assumptions about representational schemes. Marcus’ informal observations suggest that, when people are presented with sentences such as: \textit{a rose is a rose, a tulip is a tulip, a lily is a lily},

\(^{10}\)Note that, although Marcus’ discussion assumes local representations, these are not the only ones that could satisfy the training space criteria. For instance, a distributed scheme could suffice if the representation of the novel output provides a novel value to at least one node that has not varied during training. If an argument can be made that such a scheme is likely to be the case, Marcus does not provide it.
they respond to the novel test case: a blicket is a ?, with the word blicket. In order to conclude that humans are generalizing beyond the training space, their internal representation of blicket must include a novel value at a node that has not varied during training. One way this might occur is if each word were represented by a single node. Then, the node for blicket would be consistently inactive during training. But it would be activated, hence have a novel value, during testing. Much of Marcus’ discussion and demonstration of how Elman’s Simple Recurrent Networks (SRN’s) fail on this task do clearly assume this representational scheme [52, 53, p. 50]. Yet Marcus gives no reason why such a scheme would be likely in humans. In fact, he acknowledges that this option, where an entire word is represented by a single node, is probably not the case in humans. He states that Elman is likely correct in suggesting that “novel words aren’t represented as novel perceptual dimensions” [54].

Another alternative is a representational scheme in which each word is represented by a set of nodes, where each node corresponds to a single grapheme (or phoneme). In this scheme, the test output blicket, might fall outside of the training space by virtue of the novelty of its constituents relative to the training set. For example, if the sentences containing rose, tulip, and lily are the only training cases, then the individual nodes representing the graphemes b, c, k would not have been activated by the training set11. The test output blicket would then seem to lie outside the training space, since it includes novel activation values at nodes that had not varied for the training cases.

Marcus’ discussion of the Sentence Completion task does not include arguments for such a representational scheme in humans. But for this example, and in his formal studies [8, 56], he considers a related way in which humans might represent linguistic data. Since phonemes can be viewed as a composite of finer elements called phonetic features12, Marcus considers a scheme in which each node represents a single phonetic feature.

The Sentence Completion example illustrates how difficulties arise with Marcus’ problem formulation under any of these schemes for representing linguistic data (i.e. one grapheme,
CHAPTER 3. ANALYSIS

phoneme or phonetic feature per node). For instance, humans who perform the Sentence Completion task are likely to have considerable previous experience with all of the graphemes (and phonemes and phonetic features) used. So blicket would be a novel combination of known features. Considering this, it is far from clear that any node representing, for example, the grapheme b, is being activated or varied for the very first time by the test word blicket. In order for blicket to be novel in the sense that Marcus intends, he would need to explain why it necessarily provides a novel value to a node that has not varied during training. So even if Marcus could provide reason to believe that humans are using one of these internal representational schemes (i.e. one grapheme or phoneme or phonetic feature per node), it would not be sufficient to support his claim that humans are generalizing beyond the training space, since assumptions about previous learning in humans would also need to be taken into account. That is, the Sentence Completion task which is set up as a challenge for modellers might not be directly comparable to the one that humans are performing. Humans may actually be generalizing from more than just the immediately apparent training set — possibly even from one that would not require exceeding the space of their actual training set.

Although Marcus does not include it as part of his definition of a training space, he does observe in the context of his simulations that “What matters is the training space with respect to some particular function” [52]. That is, his simulations show that previous training of every node on one function cannot help the network generalize beyond the training space on a different function. Marcus might attempt to argue that the previous training of humans does not undermine his conclusions about the Sentence Completion task, since they have not encountered the relevant function before. Evaluating this argument would require consensus on what the relevant function is.

---

12Phonetic features, sometimes called distinctive features [3, pp. 340-341], are the constituents of phonemes. Phonetic features are not to be confused with features in the pattern recognition sense of the term, which define the dimensions of a feature space (i.e. a representational space). A phonetic feature may or may not correspond to a feature, (hence to a dimension of the feature space), depending on the particular representational scheme involved. In Marcus’ scheme of one phonetic feature per node, there is such a correspondence. Phonetic features themselves are considered to be “complex, temporally varying patterns of frequencies” [3, p. 341].
CHAPTER 3. ANALYSIS

In what sense does the previous training of humans involve a ‘different function’ from the one they are using to perform the Sentence Completion task? For the networks of concern to Marcus, the relevant ‘function’ might be taken to be the task as a whole, which requires completing sentences of the form $A(x)$ is $a(x)$. Then, the elements ($b$, $c$, and $k$) in blicket would be novel relative to the function, since they have not appeared in any sentence of this form as part of the network’s training set.

However, there is reason to believe that learning (i.e. training the weights of a network) is not the immediate mechanism involved when humans perform the Sentence Completion task (for details, see Chapter 4, Section 4.2.2). If this is the case, then the Sentence Completion task in humans would depend on skills that they have acquired prior to the presentation of the sample sentences ($a$ rose is a rose, etc.). So in humans, previous learning would be very relevant. If this previous learning involves encountering and completing sentences of the form $A(x)$ is $a(x)$, then Marcus would still need to consider what the actual training set would be, and give reasons why, for example, some particular phonetic feature of $b$, $c$, or $k$ would not have been varied in that set. Alternatively, if the previous learning does not include encountering and completing sentences of the form $A(x)$ is $a(x)$, then there must be one or more other previously-learned ‘functions’ that are involved when humans perform the task. Marcus would need to explain why blicket is not just a novel combination of features that have varied in the training set of such ‘different’ but apparently relevant-in-humans functions.\(^\text{13}\)

For the reasons outlined in this section, Marcus’ claim that the Sentence Completion task shows humans generalizing beyond the training space calls for more support than it might appear to at first glance, and more than he has offered so far.

\(^\text{13}\)This is not to argue that models which exhibit training independence would necessarily be able to solve the problem in the training spaces humans might be using. (For example, see Hadley and Vilcu [39] for evidence that, contrary to Elman [23] some kinds of previous training do not seem to help Elman’s Simple Recurrent Networks [20] to generalize Marcus’ infant grammar tasks [56].) The argument that is being made in this thesis is that Marcus’ description of human generalization in terms of a training space is problematic without more serious consideration of what their actual training set and internal representations are likely to be.
CHAPTER 3. ANALYSIS

Formal Studies of Human Subjects

Although the term training space is not used in the infant study [56], pains are taken to choose test items that are novel in particular ways. The study included three experiments. In the first experiment, test cases were carefully chosen so that they were not composed of words that had been used during the habituation (or ‘training’) phase. For example, infants were habituated to phrases like ga na na (ABB) or li ti li (ABA), then tested on phrases like wo fe fe or wo fe wo. The experiment tested for generalization to words which, as a whole, were novel relative to the training set. But Marcus states that it did not test for generalization to novel phonetic features (e.g. ± voiced) that had not varied during training [56]. For this reason, the authors include a second experiment in their study, where test items are considered to contain novel phonetic features. For example, the test items (e.g. ko ga ga) contains words with both voiced and unvoiced consonants, whereas all the training items (e.g. wi li wi) are voiced [56].

The connection to the issue of training space is not made explicit in the infant study, but the choice of training and test sets suggests that the researchers are assuming that words are represented in terms of phonetic features, where at least one node is unique to each phonetic feature (or feature value). In this way, a test item such as ko ga ga (i.e. with both voiced and unvoiced consonants) would be outside the training space of a habituation set in which all consonants were voiced. Although their representational assumption is neither supported nor acknowledged, Marcus later cites the infant study as evidence that infants can “freely generalize”14 [53, p. 38]. More specifically, he later states that “these finding suggested that learners can generalize the artificial grammar beyond the training space”[8].

If the test items are considered to exceed the training space, it would be due to the novel feature values they contain. For example, in the second and third experiments of the study, only voiced phonemes are included during habituation, whereas the test cases include also unvoiced phonemes. So the unvoiced phoneme in the test case would be responsible for providing a novel value to at least one node that had not varied during training. But if the infants had heard consonants such as b (voiced) and p (unvoiced) before the experiment, then the feature

14Marcus appears to use the term “free generalization” [53] as a synonym for Universal Generalization.
± voiced would have varied prior to presentation of the novel test case. The authors of the study [56] do not discuss whether the infants could have acquired relevant experience prior to the experiment. They do not argue that the test cases include individual phonemes, phonetic features, or feature values that the infants have never previously encountered. Indeed, it seems difficult to imagine how seven-month-old infants, most of whom, presumably, have been exposed to a considerable amount of spoken English, would not have previously heard both voiced and unvoiced consonants in many different contexts.

If we consider that even infants may have previously encountered and perhaps even learned to distinguish\textsuperscript{15} all of the phonemes in the study, then the actual training set may be more extensive than the just sample phrases presented in the study. Showing that a network must exceed the training space to perform the task (which, presumably cannot be comparable to the human task unless it allows access to the same data set) would require establishing that the actual data available are insufficient to exercise each node of the network. If, as seems likely, the infants are familiar with the phonetic features of English, then each test item would be a novel combination of known phonetic features, but would not necessarily provide a novel value to a node that has not varied during training.

Marcus might argue that the feature unvoiced only needs to be novel relative to the particular task in the study. Particularly since the task is not clearly one of generation, and, as the experimenters themselves acknowledge [56] (see Section 3.1.1), might be accomplished by comparing pairs of syllables, such an argument should give at least some consideration to why the infants should be assumed to have had no previous experience on the task of distinguishing syllables containing voiced and unvoiced consonants.

Marcus might consider retreating to a position that argues: ‘if infants were using the architectures of concern to him, then they would be generalizing outside the training space; but since those architectures cannot exceed the training space, the infants must not be using them’. Such an argument would require showing why the networks must exceed the training space in

\textsuperscript{15}As early as about one month of age, infants can distinguish between the syllables \emph{ba} and \emph{ma}, as measured by changes in the infants sucking rate during habituation experiments [61, pp. 401-402].
order to perform the generalization task the infants are doing. It would still need to justify the assumptions about the types of representations used and whether the infants are generating specific internal predictions (see Section 3.1.1).

One way in which even a network might not need to exceed the training space would be if the data were represented using a sonority scale (a measure of vowel likeness). Shultz and Bale [68] have recently suggested that such a representational scheme could explain how the infants could generalize to the novel cases in this particular study\textsuperscript{16}. Using a sonority scale, the novel test items do not lie beyond the training space of the training set specified in the study. In Shultz’ and Bale’s representational scheme, each consonant or vowel is represented by a single (integer) sonority value between -6 and 6. For example, \textit{ga ti ga} would be represented at the six-node input layer by the vector (5 6 -6 4 -5 6). Shultz and Bale report that test items were novel in the sense that they exceeded the training range [68]. They consider an item (e.g. a consonant or a vowel) to exceed the training range when it presents a value (to some node) that is outside of the extremes of the range of values which have been presented at that node during training. For example, a test item would present a value of -6 at a node which, during training, had only ever encountered values between -5 and -1.

But note that exceeding the training range is not the same as generalizing outside the training space as Marcus defines it. Although Shultz and Bale do not observe so themselves, it can be seen from their input data that no test items exceeds the training space. That is, no test item presents a new value to a node that had not varied during training. So given the possibility of representing the data using a sonority scale, even networks would not need to exceed the training space.

This example illustrates how our understanding of observed human behaviour can be crucially altered by our assumptions about input representations. It calls into question Marcus’ claim that infants are generalizing beyond their training space. This weakens his larger

\textsuperscript{16} Vilcu and Hadley [78] have shown that, while the network’s performance might be consistent with the particular test cases reported in infant study, it does not produce the kind of responses one might expect from the infants for other novel test items — even some that exceed neither the training space nor the training range.
argument and reveals an important difficulty with representation and the criterion for novelty in his formulation of the Universal Generalization problem.

Compared to the infant study, the researchers in the study of Hebrew word formation [8] address the training space question more explicitly. Instead of studying generalization of an artificial grammar, as in the infant study, they propose investigating the scope of generalization in natural language.

Recall that the word formation study investigates how a particular constraint on the formation of Hebrew words can be generalized to novel items. The constraint disallows words with a root structure of the form XXY (i.e. where the first two syllables of the word root begin with identical consonants). Identical consonants at the beginning of two adjacent syllables in the word root are called *geminates*. So the constraint disallows geminates at the start of Hebrew words.

The subjects in this study are native Hebrew speakers who are also fluent in English. In order to test whether subjects can generalize the constraint on geminates beyond the training space, the researchers consider four phonemes (i.e. *cb, j, w, th*) which are common in English but absent in Hebrew. Because these phonemes are not part of the Hebrew language, the researchers reason that subjects would not have used these phonemes in geminates prior to the study [8]. The study uses these four phonemes as the novel items with which to test the scope of the subjects' generalization of the constraint.

However, three of these novel phonemes (*cb, j, w*) can be perceived to be composed of features that exist in Hebrew, as the researchers acknowledge [8]. This leaves the single phoneme, *th*, as possible evidence for generalization beyond the training space. This phoneme includes a distinctive feature called Tongue Tip Constriction Area (TTCA) [26]. The values *narrow* and *mid* of the TTCA feature are present in various phonemes used in Hebrew, whereas the value *wide* is present only in English. According to the researchers, this means that the phoneme *th* (with TTCA=*wide*) falls outside of the training space.

Berent et al. do acknowledge that their conclusion (i.e. that *th* is outside the training space) depends on assumptions about the subjects' internal representations of phonemes. They argue
that separation of phonemes into distinctive features is psychologically plausible, since a variety of linguistic phenomena are best explained by them. Specifically, they cite theoretical and empirical evidence from linguistics for their view that the distinctive feature TTCA does correspond to a dimension of the subjects' feature space and that the representation of the phoneme th includes a feature value (i.e. TTCA=wide) which is foreign to Hebrew [8,18].

To conclude, as the researchers do, that the phoneme th lies beyond the subjects' training space would require knowing that the novel feature value TTCA=wide activates at least one node that has not varied during training on the task at hand. But TTCA is a feature that takes on three possible values: narrow, mid, and wide. Given that two values (narrow and mid) of the feature TTCA do appear in the Hebrew language, (hence in the presumed training data), TTCA appears to be a feature that does vary in the training set. So if humans are using the distinctive feature TTCA as a dimension of their feature space (i.e. representing it by a separate input node), then the novel feature value (TTCA=wide) might require some extrapolation within the training space, but it would not lie outside the training space.

In order to conclude that the novel feature value (TTCA=wide) is outside the training space, Berent et al. might, but do not, give reasons to believe that an alternative representational scheme is being used by humans. Here, the dimensions (i.e. nodes) of the feature space would correspond to feature values (e.g. TTCA=wide), rather than to distinctive features (e.g. TTCA). Then, the novel feature value (TTCA=wide) would activate at least one node that has not varied during training, and hence, would fall outside the training space.

The Hebrew word formation study presents only this single example (th) as a case of humans generalizing outside of the training space. But, as shown above, the observations of this critical case could also be consistent with generalization within the training space, depending on how subjects internally represent the data. Since important representational details that would be necessary to establish its conclusions about the training space are overlooked, the word formation study does not clearly establish that humans are generalizing beyond the training space.
Additional considerations, which the researchers do not address, are the intricacies of the dynamic frequency patterns which compose the distinctive features. It is not immediately obvious whether representations that might be available to subjects at this level would be consistent with the researchers' assumptions about feature separation.

**Implications**

Marcus' claim that humans perform Universal Generalization depends on whether humans can be seen to exceed their training space. The training space is defined relative to the representation (i.e. how data are mapped to nodes), the training set, and perhaps the task. In humans, we cannot directly observe how data are mapped to nodes (i.e. neurons or groups of neurons). It is also difficult to determine whether, or in what task context, human subjects might have previously encountered the test items.

The discussion in this section has described representational schemes in which the test data from Marcus' formal studies would not lie beyond the training space (for either humans or networks even relative to just the specific training set and task). This would be the case in the Hebrew word formation study [8] if each phonetic feature were represented by a separate node (or group of nodes). In the infant study [56], a sonority scale would have this result.

Similar unresolved questions regarding representation and previous experience arise in Marcus' informal observations of the Binary Identity and Sentence Completion tasks. In addition, there is reason to believe that the Sentence Completion task may not involve learning (i.e. weight change) as the immediate mechanism (see Section 4.2.2.). This would mean that humans must be using skills they have acquired prior to the presentation of the sample sentences. Concluding that humans are exceeding the training space would require a discussion of what the relevant previous learning would be and whether the test cases exceed the training space of the actual training set humans are using.

These difficulties concerning the novelty of test items, in combination with the discrepancies between the forms of the tasks presented to humans and artificial networks (Section 3.1.1), raise doubts about Marcus' claims that humans perform Universal
CHAPTER 3. ANALYSIS

Generalization. He argues that, since humans do appear to perform Universal Generalization, but current eliminativist models cannot, symbol manipulation seems to be a better explanation than eliminativism [52, 53]. While his conclusion about the role of symbol manipulation in cognition may well be correct, his argument for it is questionable, since his first premise has not been clearly established. Inconsistencies and unstated assumptions in his formulation of the problem mean that the tasks on which the networks fail are not necessarily equivalent to those performed by humans.

3.1.3 Evidence for Type-2 Generalization

Clark and Thornton claim that Type-2 problems are frequently solved by “real learners” in “biologically realistic settings” [15]. They suggest that such problems are “rife in biologically realistic settings and in domains ranging from simple animat (simulated animal or autonomous robot) behaviours to language acquisition” [15]. But their description of tasks in these domains discusses only artificial neural networks and machine learning algorithms. Evidence or arguments that actual humans or animals must be using representations that satisfy the Type-2 criteria are not presented.

In the domain of animate behaviour, Clark and Thornton describe the task of “conditional approach”, which consists of the following:

- when no object is sensed, swivel right 10 degrees
- when an object is sensed that is distant, or small but nearby, move forward (and in some cases, change direction)
- when a large object is sensed that is nearby, do not move.

In their simulation, the two outputs to be learned represent the amount of drive applied to the two wheels of the animat. This depends on the ratio (i.e. relation) between the inputs of apparent closeness and apparent width. The inputs and outputs are represented as real numbers in the range 0.0 to 1.0 [17]. The researchers’ simulations showed that feedforward
artificial networks based on back-propagation, as well as standard symbolic machine learning algorithms, such as ID3 [65] and C4.5 [66], were unsuccessful on this task.

But no argument or evidence is given that any biological system senses or represents object location by these two dimensions (i.e. closeness and width). Even the form of the outputs, as two real numbers which determine the drive of two wheels, would require justification before the task could be clearly associated with actual animal behaviour.

In the linguistic domain, Clark and Thornton discuss the language acquisition task performed by Elman's [20] Grammar Learning network. They describe this to illustrate that incremental learning could be a useful technique for finding appropriate recodings. Elman's task is considered to be Type-2 because it requires learning relations (cross-clausal agreement) between inputs (verb/subject number agreement) that are not apparent until learning shorter-range dependencies has provided a recoding of the inputs. It may be that in humans, language acquisition involves solving a similar task, with inputs that make it Type-2, but Clark and Thornton do not discuss how Elman's task might compare with tasks performed by humans.

3.2 Issues in Evaluating Problem Equivalence

The preceding discussion of Universal Generalization in humans reveals how a number of complex and subtle issues about representation and task form arise when comparing generalization tasks.

3.2.1 Representation

Representation is important in generalization because it can potentially change the class of a generalization task. The two main formulations\(^ {18}\) of the research problem — Marcus' 

\(^{17}\)No data set is reported for the example they discuss.

\(^{18}\)Systematicity (Section 2.3.1) also presents an important and very productive research challenge. But since its formulations to date [24, 32, 60] are less explicit at the level of representation and task form than are those by Marcus [52, 53] or Clark and Thornton [15, 73], the latter problem formulations are the primary focus of this analysis. Systematicity and the other related problems that were identified in Chapter 2 (Section 2.3) are revisited in Chapter 4.
Universal Generalization [52, 53] and Clark’s and Thornton’s Type-2 Generalization [15] — present specific criteria for classifying generalization tasks. These criteria depend intimately on representation — to the point where a change in representation could mean a change in the classification of a task. For example, the geometric separability criteria of Type-2 generalization — a measure of how data cluster within the input space — cannot be evaluated without knowledge of how data are represented. In the case of symbolic machine learning algorithms, the features that are used to represent the data determine the dimensions of the input space. In the case of neural networks, the dimensions of the input space are determined by the way in which the data are mapped to the nodes of a network (see Section 2.1.2). Without knowing the dimensions of the input space, it is impossible to measure the GSI (i.e. the clustering of points in the input space), since both location and distance are undefined (see Section 2.2.2).

The authors of both main formulations acknowledge that the details of representation are crucial. Marcus acknowledges that his conclusions about the limitations of neural networks that exhibit training independence apply in the context of a fixed representational scheme [53, p. 177]. And one of Clark’s and Thornton’s main points is that a Type-2 problem might be reducible to a more tractable Type-1 problem by means of representational recoding [15].

In concrete terms, this means that details about how individual data items are presented to human subjects and networks need to be considered carefully when evaluating any argument that depends on task equivalence. For example, Marcus presents the Binary Identity task (Table 2.1 on page 6) as an example of a Universal Generalization task that humans perform well. When Marcus argues that a back-propagating multilayer perceptron cannot perform this task, the first input is presented to the network as the vector (1, 0, 1, 0). Is this equivalent to what a human takes as ‘input’? In what sense is your perception or conception of ‘0’ comparable to an inactive neuron (or group of neurons)? How would we know this? Knowing how humans represent data internally is pivotal when discussing whether Marcus’ definition of Universal Generalization applies. Universal Generalization, by definition, requires generalization beyond the training space. But, as discussed in Sections 2.1.2 and 3.1.2, this
requires knowing how data are represented, since the dimensions of the training space are
directly defined by how data are mapped to nodes.

Even if equivalence of individual items could somehow be established, sensory modality
and order of presentation also affect problem equivalence. For example, we might not initially
be aware that viewing the data in Marcus' identity task can give us sensory information and
cues about ordering and chunking that may not be taken into account in the corresponding
problem formulated for artificial networks. Marcus would present the identity task data in
sequential chunks to a network, whereas a visual presentation for humans permits additional
possibilities for re-ordered, repeated, or parallel perception of the data elements. Consider
trying to respond to the test question if all the data had been presented verbally, rather than in
writing. Or imagine that the data had been presented as an uninterrupted string of 1's and 0's.
Would the task of predicting the next four digits be equivalent to the original one?

Add to this the rich semantic understanding most adults have of the symbols on the page.
Many would know how to read the strings as multi-digit decimal or binary numbers and could
recognize that mathematical properties, such as oddness, parity, or simply numeric value, are
preserved from input to output. Do we know that none of this potentially available
information is included as part of the internal representation of task? The definition of
Universal Generalization depends on whether the training space is exceeded. But if we ignore
the possibility that humans have access to a richer relevant training set than is immediately
apparent, we may falsely conclude that humans are exceeding their training space.

3.2.2 Task Form

Additional difficulties arise when behavioural responses must be interpreted as the outputs of
a generalizing biological network. In Marcus' formulation, the difference between generation
and classification is an important factor in determining whether or not a task is a case of
Universal Generalization. Universal Generalization requires that the test output is novel
relative to the training data (i.e. outside the training space). So novelty depends on how the
data are represented and on what information is available as part of the training set. Since we
cannot isolate or directly observe the history and activity of the relevant biological networks, many interrelated assumptions are implicated.

Even when tasks are compared only within the context of artificial neural networks, questions about availability and presentation of data arise. For example, does translating a generalization task from a format that could be presented to a supervised network to a format suitable for an unsupervised network mean that the two tasks are not equivalent? A goal of this thesis is to provide a notation and terminology to support a more lucid discussion of such complex, interrelated issues. The following section introduces such a framework.

### 3.3 Framework for Task Specification

The literature reviewed in Chapter 2 presents a challenge for research into neural networks by identifying types of generalization *tasks* that humans seem to perform more readily than do many neural networks. The analysis of the current chapter has shown how fundamental issues concerning representation and task form arise when comparing generalization tasks. In order to support a clearer discussion of disparate problem formulations and contexts, a template for the specification of a generalization task is introduced in Section 3.3.1. Section 3.3.2 provides terminology for distinguishing two levels at which a task may be specified. The framework is used in Chapters 4 and 5 to help make assumptions about representation and task form more visible, hence accessible to analysis.

#### 3.3.1 Generalization Task Specification

Generalization typically involves observing a pattern in a set of examples (training cases) and extending that to novel items (test cases). Formally, a *Training Case* (or *Test Case*) is an ordered pair of vectors, $(\vec{x}, \vec{y})$, where for $p, q, \geq 1$,

\[
\bar{x} = (x_1, x_2, \ldots, x_p) \text{ is the input and} \\
\bar{y} = (y_1, y_2, \ldots, y_q) \text{ is the output}
\]
A Generalization Task Specification includes the following elements:

- **A Training Set:** One or more training cases. As a default in this notational scheme, both the inputs and the outputs of the training cases are available to the generalizer. Exceptions to this should be noted as part of the task specification.

- **A Test Set:** One or more test cases. In each test case, only the input vector is available to the generalizer. In a task specification, the output vector of a test case is the response expected from a generalizer that has successfully performed the task\(^{19}\).

Table 3.2 illustrates the form of a typical data set (i.e., the training set and test set). Here, only one test case is indicated, although in general, several may be specified.

### Table 3.2: Template for Generalization Task Data

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Set</td>
<td></td>
</tr>
<tr>
<td>( \bar{x}<em>1 = (x</em>{11}, x_{12}, \ldots, x_{1p}) )</td>
<td>( \bar{y}<em>1 = (y</em>{11}, y_{12}, \ldots, y_{1q}) )</td>
</tr>
<tr>
<td>( \bar{x}<em>2 = (x</em>{21}, x_{22}, \ldots, x_{2p}) )</td>
<td>( \bar{y}<em>2 = (y</em>{21}, y_{22}, \ldots, y_{1q}) )</td>
</tr>
<tr>
<td></td>
<td>( \vdots )</td>
</tr>
<tr>
<td>Test Set</td>
<td></td>
</tr>
<tr>
<td>( \bar{x}<em>n = (x</em>{n1}, x_{n2}, \ldots, x_{np}) )</td>
<td>( \bar{y}<em>n = (y</em>{n1}, y_{n2}, \ldots, y_{nq}) )</td>
</tr>
</tbody>
</table>

A discussion of the types of elements that vectors may contain and of how these are mapped to the nodes of a network is undertaken in the following section.

---

\(^{19}\)In a task specification, the test output is a desired or target output. The actual output (response) of a generalizer is not part of the task specification.
3.3.2 Coarse and Ground Form Specifications

Because generalization tasks are presented to both humans and symbolic algorithms, as well as directly to neural networks, the framework introduces terminology to distinguish between two forms or levels of task specification: Coarse and Ground.

Figure 3.1: Mapping of Data to Nodes of a Network

A ground form task specification is one in which each vector element is mapped to a node in a one-to-one way. Figure 3.1 shows how, for example, training pair \((\overline{x}_2, \overline{y}_2)\) from Table 3.2 would be made available to a multilayer perceptron. Each element, \(x_{2i}\), of the input vector, \(\overline{x}_2 = (x_{21}, x_{22}, \ldots, x_{2p})\), would be presented to a separate input node. Each element, \(y_{2j}\), of the output vector, \(\overline{y}_2 = (y_{21}, y_{22}, \ldots, y_{2q})\), would be used by the weight update rule as a target value for an individual node of the output layer.

In a ground specification, vector elements necessarily take on only numeric values, since a single node can only have numeric activation values. These may be continuous or discrete, including binary\(^{20}\). Ground specifications describe the data at a level suitable for presentation to a network, since the vector elements indicate the activation values of individual nodes.

Coarse form specifications, on the other hand, present the data at a more abstract level than do ground specifications. Task descriptions in which the mapping of vector elements to the

---

\(^{20}\)Typically, activation values of neurons range from 0 to 1. In some models, is [-1,1] is the range [42, p. 11].
nodes of a network is either unspecified or not one-to-one are coarse form specifications. In a coarse form specification, vector elements may contain non-numeric data, such as letters, digits, words, or graphical symbols. These may correspond to items that are presented to humans at a sensory level. They may be the actual visual symbols that are viewed or merely a written representation of spoken words. For example, in a description of the habituation task in Marcus' infant grammar study [56], vector elements might be letters or syllables (e.g. /ti ti ti/), but the data the infants actually encounter is a sequence of sounds.

Coarse specifications can be viewed as a kind of shorthand for one or more ground form specifications. In order to determine which possible ground form specifications might be considered to be comparable interpretations of a coarse specification, assumptions about representation (i.e. the mapping of data to the nodes of a network) must be made explicit. For example, the argument by Berent et al. [8] (Section 3.1.2) that humans likely represent words by means of phonetic features, such as TTCA, can be seen as an attempt to establish a connection between the written or spoken words presented to the subjects (i.e. the data of a coarse specification) and the activation values of nodes of an internal network (i.e. the data of a comparable ground specification).

The usefulness of the framework introduced here is further illustrated by its application in Chapters 4 and 5. Section 4.3.1 shows how failing to distinguish between coarse and ground forms of a task specification can lead to unwarranted conclusions about task equivalence. In Sections 5.1.1 and 5.2.2, the framework is applied in detailed discussions of sample generalization tasks and solutions.

3.4 Summary

Previous formulations of the research problem have been considered meaningful largely because they can be used to describe and explore differences in how humans and artificial neural networks can generalize. Such comparisons play an important role in arguments about the role of symbol manipulation in cognition (e.g. [24, 52, 53]).
Comparing how humans and artificial networks generalize depends on being able to observe how they perform differently on comparable tasks. The analysis undertaken in this chapter reveals ways in which the meaningfulness of previous problem formulations as descriptions of human generalization is limited by inconsistencies, and unacknowledged or unsupported assumptions about task equivalence. In particular, this chapter identifies fundamental issues regarding representation and task form and shows how they arise in the two main previous problem formulations — Universal Generalization [52, 53] and Type-2 Generalization [15].

The formal empirical evidence Marcus presents for Universal Generalization in humans [56, 8] is weakened by discrepancies between the form of tasks performed by subjects and those which are shown to confound eliminativist networks. Tasks in his formal studies of humans do not clearly satisfy the UQOTOM criterion for Universal Generalization — which requires generating a unique novel response (output) related to its input by a universally quantified one-to-one mapping. Rather, subjects are only observed to discriminate/classify inputs into a small number of categories. Here, the UQOTOM is between pairs of inputs, but not between inputs and outputs.

Issues of representation arise when considering whether human generalization satisfies the second criterion of Universal Generalization: that novel items must exceed the training space by providing a novel value to at least one node that has not varied during training. This is particularly difficult to establish in humans, since we currently cannot observe how the data are represented in humans. Depending on the representational scheme used, neither humans nor machines would necessarily need to exceed the training space. Additional difficulties arise in establishing what the relevant training set (hence, training space) is for humans, since it is unclear what previous relevant experience they may have had.

Although the limitations of Marcus' problem formulation do have implications for the validity of his larger argument about symbol manipulation, this does not mean that all of his conclusions are necessarily incorrect. For instance, the validity of his claims that standard eliminativist methods fail to perform Universal Generalization as he defines it, is not contradicted by the weakening of his arguments that humans can do so. However, the significance
of this failure of eliminativist methods, and thus the *meaningfulness* of his problem formulation, do depend largely on whether Universal Generalization accurately describes what human do.

Similarly, Clark and Thornton suggest that their formulation of Type-2 Generalization describes problems that are regularly solved in "biologically realistic settings" [15], yet they offer no evidence or arguments that actual humans or animals must be using representations that satisfy the Type-2 criteria. If they mean to establish that biological learners perform Type-2 generalization, representation is crucial, since one of their main points is that a change in representation can change a problem from Type-2 to Type-1.

In this area of research, the 'objects' of study and classification are individual generalization tasks. Their intangibility makes it especially important to have a clear and consistent terminology and notation for describing them. Previously, researchers have used a variety of informal notations and natural language to describe generalization tasks. In order to support a clearer discussion of task equivalence, this chapter introduces a notational framework and terminology for the specification of generalization tasks. The framework helps to make assumptions about representation and task form more explicit (hence visible, and so accessible to analysis) by requiring explicit specification, in a standard format, of:

1. what information is available,

2. how it is represented (i.e. mapped to the nodes of the network), and

3. what response must be produced.

Due to current limitations of neural imaging technology, we cannot observe directly how humans represent information internally, so a generalization task that is performed by humans must necessarily be described at a *coarse* level\(^{21}\). In order to determine comparable tasks which could be presented to artificial networks (i.e. possible ground form specifications), the realistic possibilities for mapping the data to the nodes of a network must be considered explicitly. The framework is used in the remainder of this thesis as a tool to for exploring such possibilities.

\(^{21}\)By definition, a task specification in which the mapping of data to nodes is unknown is a *coarse* level specification (Section 3.3.2).
Chapter 4

Reformulation and Classification

This chapter considers how the insights of previous researchers might be integrated into a broader understanding of different types of generalization. The classification scheme proposed in this chapter uses the notational framework that was introduced in Chapter 3 to support the comparison of tasks from disparate problem formulations and contexts. Grounds for generalization is identified as a fundamental distinguishing characteristic of the problem classes.

This chapter addresses parts ii) iii) and iv) of the thesis problem statement.

ii) How do the formulations relate to each other?

iii) What underlies them?

iv) Can the formulations be improved or even unified?

Section 4.1 explores question ii) by considering how aspects of previous formulations converge, diverge, or coincide with each other. Section 4.2 addresses question iii) by identifying key factors involving grounds for generalization, similarity, and the role of learning, which can be seen to underlie distinctions between problem classes. Based on these, a more unified understanding of the problem space is possible. This new classification scheme addresses component iv) of the thesis problem statement. Sections 4.3 and 4.4 present descriptions of the two main classes of generalization, their subclasses, and sample tasks. The chapter concludes with a summary in Section 4.5.
4.1 Common Ground

This section considers some of the ways in which previous formulations of the research problem can be seen in relation to each other. Each formulation can be considered to be a classification scheme, in that it distinguishes between types of tasks that can be solved (or not) by various types of generalizers. Although there is considerable variation in the factors considered by researchers to be important in the classification of a task, they do naturally encounter a number of common issues and offer some related insights. These are considered in Sections 4.1.1 and 4.1.2.

4.1.1 Universal Generalization and Type-2 Generalization

A comparison of the two main formulations — Universal Generalization and Type-2 Generalization — shows significant differences in how factors such as relations, task form, novelty, and representation are used to classify generalization tasks. Although Marcus [53], as well as Clark and Thornton [15], observe that the generalization of relations is a key characteristic of the problems of interest, their two formulations are not entirely equivalent in this respect. Marcus focuses on generalizing relations (specifically, UQOTOM) between inputs and outputs. Clark and Thornton are concerned with generalizations where each output depends on the relationship between (elements of) the inputs. Closely related to this is a difference in the form of tasks encountered in the two schemes. Marcus focuses on generation tasks, whereas Clark's and Thornton's classification scheme applies only to binary classification tasks. So the input-output relations in Type-2 generalization are not what Marcus would call UQOTOM; rather, they are many-to-one mappings. This distinction between the two formulations is somewhat blurred by the inconsistencies between Marcus' definition of Universal Generalization and the types of examples he provides. As discussed in Section 3.1.1, the tasks in his formal studies do not clearly satisfy the UQOTOM criteria of Universal generalization. As well, some of Marcus' own examples (i.e. concatenation, multiplication) of UQOTOM are actually many-to-one mappings. Marcus acknowledges that concatenation is not strictly one-to-one, since a given string
could be formed by concatenating substrings in more than one way: (e.g. "ab + cd = abc + d = abcd"). He suggests considering instead the "one-place predicates (concatenate with -ed; concatenate with -ing etc.), each of which individually would be one-to-one" [53, pp. 176-177]. Marcus identifies multiplication as another example of a UQOTOM [53, p. 45], but does not observe that similar arguments would apply (e.g. 2 * 6 = 3 * 4 = 12).

Another factor which differs between the two formulations is the scope of generalization. Type-2 Generalization extends to test cases that are novel in the sense that they contain previously unseen inputs (i.e. those which were not included in the training data). Universal Generalization describes a much more specific and subtle form of novelty. Specifically, Universal Generalization extends beyond the training space (i.e. test items contain a novel value in a feature that has not varied during training).

The authors of both formulations claim to describe a class of generalization that is beyond the scope of back-propagating multilayer perceptrons. They do naturally encounter a number of common issues and offer some related insights. They both observe the importance of relations. Both formulations characterize tasks in terms of factors (GSI, training space) that depend on how a task is represented. And the authors of both acknowledge that a change in representation could alter how a given task is classified (see Section 3.2.1). But the disparity between the two formulations is not merely a difference in approach. Due particularly to the fact that they apply to different forms of tasks, Universal Generalization and Type-2 Generalization appear to describe separate classes of tasks.

### 4.1.2 Related Problem Formulations

Universal generalization seems to have more in common with some of the related problems reviewed in Chapter 2 than with Type-2 Generalization. For example, Phillips [63] considers his own Cross-Category and Non-Category generalization challenges to be cases of Universal Generalization. Phillips [63] also suggests that Strong Systematicity is "closely related" to Universal Generalization, and claims that training independence is an obstacle for both (see Section 2.3). In his commentary [21] on The Algebraic Mind, Elman is more precise about how
the two problems might be related. He suggests that systematicity — or at least, an element of strong syntactic systematicity (i.e. generalizing to words that appear in novel syntactic positions) — is comparable to Universal Generalization in the sense that it can be interpreted as a case of generalizing outside the training space [21]. But in Marcus’ response [54], his disagreement is clear: Universal Generalization extends to novel items that have not appeared anywhere in the training data.

This disagreement can be seen to depend, in part, on whether one considers the novel test 'item' to be a word which has been encountered during training, or a sentence which has not been encountered during training. A sentence containing a previously-encountered word in a novel syntactic position would be novel in the sense that it is a novel combination of known words. So the disagreement illustrates how fundamental confusion can arise due to a lack of consistent notation and terminology. Further, whether a novel item is represented as a combination of previously-seen elements is also crucial, as can be seen from the analysis of several of Marcus’ sample tasks (see Section 3.1.2). For example, in the Sentence Completion task, the test word blicket may be novel in that it has never been encountered anywhere in the training set, as Marcus requires. But if blicket is represented as a combination of previously-encountered elements — e.g. letters, graphemes, phonemes, or phonetic features, each of which is now appearing in a novel context (i.e. position) — then it does not necessarily fall outside the training space, and so does not satisfy the definition of Universal Generalization.

Hadley [35] observes that his own formulation of Systematicity [32] differs from Marcus’ [52] challenge of generalizing outside the training space, but that one of Niklasson’s and van Gelder’s [60] levels of generalization (i.e. Level 3) “appears equivalent” to Marcus’ challenge. Here, “the test sentences contain at least one atomic constituent which did not appear anywhere in the training set” [60].

1 Depending on the representation used, the sentence might also be novel in the sense that it presents a novel value to a feature that has not varied during training. For example, if each position corresponds to a separate set of nodes, then the novel word might exercise one or more nodes that did not vary in value during training.
Rather than comparing how humans and artificial networks perform, Goldstone et al. [28] describe experiments that compare different ways in which humans can generalize (see Section 2.3.3). Their observations suggest that relations may play an important role in the flexibility of human cognition and may be involved in a distinction between two types of cognitive capacities in humans.

4.2 Underlying Distinctions

Previous researchers have contributed a number of insights about how humans, artificial neural networks, and symbolic learners generalize. These include observations about relations [15, 53], novelty [15, 53, 32, 60], similarity [28, 63], representation [15, 53], inductive justification [15], and task form [53]. The specific ways in which researchers define, understand and use these factors to classify generalization tasks is quite varied and frequently contradictory — at times, it is even inconsistent within a given problem formulation2.

Despite the variation among previous formulations of the problem, researchers seem to share a common intuition that some fundamental distinction can be made between different types of generalization. This section considers what might underlie that intuition. Two key factors are identified: Grounds for Generalization and the Role of Learning. Sections 4.2.1 and 4.2.2 introduce these elements.

4.2.1 Grounds for Generalization

An examination of sample generalization tasks from the literature reveals a number of ways in which a generalization may be justified. Each involves observing a particular pattern of similarity present in the training data, then extending that pattern to novel test items. By considering the type and pattern of similarity involved, two main forms of grounds for generalization can be distinguished. These form the foundation for the distinction between First-Order and Second-Order Generalization. Depending on how these patterns of similarity are encountered within the

---

2 For example, Universal Generalization [52, 53] encounters such difficulties (Sections 3.1.1 and 3.1.2).
structure of a particular task, further subclasses can be distinguished. The details of these patterns of similarity, classes, and sample problems are described in Sections 4.3 and 4.4.

Although the terms *grounds for generalization* and *inductive justification* are used synonymously in the following classification scheme, the understanding presented here differs considerably from Clark's and Thornton's [15] probabilistic formulation of inductive justification (Section 2.2.1). The understanding presented in this thesis accommodates a broader range of ways in which test cases can be novel. For example, it has the advantage of accounting for generalization to novel outputs. Clark's and Thornton's probabilistic formulation of inductive justification does not actually support generalization to any novel outputs, since the probabilities depend entirely on how the test output has appeared previously in the training data. But a novel output, by definition, has not appeared in the training set. This means that the Type-1 and Type-2 probabilities (both unconditional, and conditional) are zero, hence no novel outputs can be justified in their scheme.

The following sections of this chapter distinguish two main classes of generalization: *First-Order generalization* (Section 4.3) and *Second-Order generalization* (Section 4.4). Within each of these two main classes, two subclasses are described. *First-Order generalization* includes the classes of *Standard First-Order generalization* (Section 4.3.1) and *Cross-Dimensional generalization* (Section 4.3.2). *Second-Order generalization* includes the classes of *Core Relational generalization* (Section 4.4.1) and a somewhat more heterogeneous class of problems, involving *Composite and Special-Purpose Relations* (Section 4.4.2).

Grounds for generalization is one of two key factors used to classify types of generalization in the scheme presented in this chapter. Before presenting the problem classes, the following section (4.2.2) describes the second key factor upon which the classification scheme is based: the role of learning.

**4.2.2 The Role of Learning**

In addition to grounds for generalization, the new classification scheme takes into account how learning is involved in different types of generalization tasks. Previous formulations of the
research problem tend to assume that learning (i.e. weight change) is the principal mechanism of generalization. Sometimes, the terms 'learning' and 'generalization' are used synonymously, or are conflated with terms such as curve-fitting, interpolation, extrapolation, and induction. A lack of clarity about how these terms relate to one another has lead to confusion and debate (e.g. [16]).

Much of the previous research concentrates on supervised learning algorithms, where the correct response (target) for each input is known (and available to the network). In this context (at least for continuous, numeric data), generalization has often been used to mean "the fitting of a smooth function to the input-output mapping, avoiding overfitting" of the training data [16]. Gradient-descent algorithms, such as back-propagation, have been considered to be quite successful learners (generalizers) in this context. But the kinds of generalization challenges examined in this thesis call for a more careful analysis of what is meant by learning and generalization.

In one broad sense of the term, a network *generalizes* well when it has *learned* (i.e. by weight modification) from the training data to produce appropriate responses to novel test data. In the case of supervised algorithms, the term 'learning' might be used to describe finding a smooth, possibly non-linear mapping, from inputs to outputs for the training data, by modifying connection strengths. Then the term 'generalization' could be reserved to describe interpolation (successful performance on test data whose values fall between those of the training data) and possibly extrapolation to test data whose values fall outside the range of values seen in the training data. This way of distinguishing learning from generalization might be useful for the

---

3 Overfitting (or overtraining) can occur in gradient-descent networks (such a back-propagation) when the network learns the examples in too much detail. One technique used to overcome this problem is to reduce the number of hidden units, so that the network is forced to abstract the essential data, rather than memorizing every instance. Another, known as cross-validation, involves dividing the data into training, validation, and test sets. The validation set is used to check the performance during training. Training is stopped when performance on the validation set stops improving, rather than continuing until the minimum error on the training data is obtained [42].

4 What is considered to be an appropriate response (i.e. output) for a particular task would be determined by the test case(s) in the specification of a generalization task (see Section 3.3). If no test outputs are included in the task specification, then the task is considered to be underspecified.
types of problems (e.g. function approximation) that are handled well by standard multilayer perceptrons\(^5\), but is not as easily applied to the types of generalization of concern in this thesis. In particular, the study of generalization to more radically novel test cases (such as those which fall outside the training space) and generalization with the types of discontinuous numeric or even symbolic data that are common in cognitive domains [34] seems to stretch the standard terminology, as does neurological evidence that unsupervised, rather than supervised learning is plausible as a biological mechanism.

In the classification scheme presented in this chapter, generalization involves \textit{learning} only when \textit{weight change} is the mechanism by which the relevant responsive capacity is acquired from examples. As is discussed in the class descriptions to follow in Sections 4.3 and 4.4, learning may or may not be immediately involved in a particular instance of generalization. \textit{Generalization means extending a pattern that is present in the training cases to novel test cases.} The classification scheme in this chapter identifies a number of distinct types of patterns that may provide the grounds for generalization (see Sections 4.3 and 4.4). Successfully performing a particular \textit{generalization task} requires producing the output(s) specified in the test case(s).

Typically, generalization tasks, such as most described by Clark and Thornton [15, 76], Marcus [52 53], and Phillips [63], are assumed to be learning problems (i.e. they involve weight modification). But not all of these necessarily are accomplished by learning. According to Phillips, the number of training cases needed to guarantee learning and generalization, even to regularly novel items, in a neural network is equal to the number of connection weights that need to be trained\(^6\). This depends on the number of nodes used to represent the input and output items and on the number of hidden layers in the network. For a fully connected

\(^5\) For example, Haykin’s [42, pp. 206-207] presentation of multilayer perceptrons makes use of the terminology in a way that is similar to this (but excludes extrapolation).

\(^6\) This result holds for networks that use linear activation functions [63]. For multilayer perceptrons, which use non-linear activation functions, the number of training cases required in practice for good generalization is \(N = O(W/\varepsilon)\), where \(W\) is the number of free parameters (i.e. weights, as well as biases, if they are used), and \(\varepsilon\) is the fraction of errors permitted [42, pp. 206-208]. So, for a network with \(n\) input nodes, \(m\) output nodes, no hidden layers, (no biases), and 10% error, this would be on the order of 10 \((n^m)\).
network with \( n \) input nodes, \( m \) output nodes, and no hidden layers, this would be \( n \times m \) weights [63]. So, for example, in Marcus' Sentence Completion tasks (Section 2.1.1), only three training cases are presented. Marcus might endorse a representational scheme where each node represents a single phoneme or phonetic feature (such as are discussed in Section 3.1.2). But with so few training cases, it is difficult to imagine that this, or perhaps any, representational scheme would allow a network to learn and generalize this task as robustly as humans seem to be able to do.

The learning assumption has also been countered by Hadley [35], who argues that humans perform generalizations, such as those described by Marcus and Phillips, in much less time than it would take for all relevant neural connection strengths to change, so learning is unlikely to be the relevant mechanism in humans in these cases. Hadley suggests that "Functionally coherent synaptic change occurs within spans of hours or days, not in a few seconds" [35]. More recent neurobiological results suggest that weight change requires approximately 50 seconds [25], but this is still longer than the time in which humans can perform tasks such as Marcus' Sentence Completion.

Other examples might be more reasonably understood as cases of sequential reasoning, than as learning. For example, Hadley [35] argues that Phillips' transverse patterning task [63] almost certainly involves reasoning by analogy and elimination. Many of Clark's and Thornton's problems [15, 76] are of a form that is comparable to their example shown in Table 2.2 on page 12. Introspection strongly suggests that some amount of conscious reasoning is involved when people solve this problem by reading the data in the table. It would be difficult

---

7 Hadley has argued that additional time would be needed for iterative transmission of the input vectors to the input layer from the memory layer: "Each vector would need at least several repeated transmissions, since they must not all be presented without interleaving the separate vectors" (personal communication).

8 Here, subjects are presented with a series of tasks which consist of three stimulus-response pairs of the form: A predicts B, B predicts C, and C predicts A — where, A, B, and C are distinct items, such as shapes or strings. Phillips claims that humans can learn the task structure from these examples and then, when a new task is presented with only one stimulus-response pair, they can predict the remaining two [63].
to imagine solving this problem without consciously subtracting the values in the second column from the first, at least for some of the training cases.

It might even be possible to perform Phillips' examples of within-category and cross-category generalization (Section 2.3.2) without any training cases. That is, a linguistically competent adult might be able to provide the correct response to John painted the chair mango. What colour is the chair? simply by virtue of understanding the test sentences. Perhaps even his non-category (aka) example could be achieved this way. Most people would be able to perform this and other tasks of the same form, even if they require previously unseen (nonsense) words as responses (i.e. non-category generalization). For example, You have a meeting in Noctao today. Where is your meeting?

4.3 First-Order Generalization

This section introduces the first main class of problems in the new classification scheme, namely, First-Order Generalization. Two forms of grounds for generalization are relevant to this class. Each involves a particular pattern of vector similarity that may be present in the task data. By examining how these two distinct patterns of similarities appear in a task, two subclasses of First-Order generalization can be distinguished: Standard First-Order Generalization and Cross-Dimensional Generalization. These are described in Sections 4.3.1 and 4.3.2.

Two inputs have vector similarity when their vector representations are nearby in representational space. A common measure of the similarity of two vectors is their inner product, since it has a straightforward trigonometric relationship to their proximity in representational space. In a nutshell, the more similar two vectors are, the closer they are in representational space, and the larger their inner product will be. This type of similarity underlies much of the learning and generalization power of many standard neural networks,

---

9 First-Order and Second-Order generalization are terms used in the new classification scheme introduced in this chapter. They do not express the same concepts as the terms Type-1 and Type-2 used in Clark's and Thornton's [15] scheme.

10 See Section 2.1.2 for a description of representational space.
since activation values are commonly computed as a weighted sum of the inputs. That is, the input to a node is the inner product of its weight vector with its input vector.

Since vector similarity is such a common and fundamental aspect of neural learning, previous researchers have naturally observed that it has relevance to generalization. For example, Phillips' [63] arguments depend heavily on his assumptions about a related property which he calls “constituent similarity.” As well, Clark's and Thornton's [73] observations about geometric separability and clustering are intimately connected to vector similarity. However, this new classification scheme distinguishes two patterns of vector similarity in terms of how they appear within the structure of a task and shows how their significance depends on the level (i.e. coarse or ground11) at which the task is specified.

4.3.1 Standard First-Order Generalization

One subclass of First-Order Generalization is Standard First-Order Generalization. This class corresponds to the type of generalization that is considered to be relatively unproblematic for standard connectionist and symbolic learning techniques. What is learned from the training set is a mapping from the input space to the output space. Here, similar input vectors (i.e. those which are neighbours in the input space) tend to be mapped to output vectors that are similar to each other (i.e. they neighbour each other in the output space). This situation is consistent with the first of Anderson’s four rules of knowledge representation: “Similar inputs from similar classes should usually produce similar representations inside the network, and should therefore be classified as belonging to the same category” [4].

Standard First-Order Generalization relies on the pattern of vector similarity described above. Generalization extends to a novel test case on the grounds that i) the relationship between the novel test input and the desired output is consistent with the learned input-output mapping, and ii) there is vector similarity between the test and training inputs and between the desired and training outputs. Generalizing to novel test cases involves applying the learned transformation to the novel test input. Depending on the details of the network and data, the

11 The terms coarse and ground are introduced in Section 3.3.2.
vector similarity (i.e. proximity in the input space) of the test input to training inputs can be used to generate an output that approximates (is near in the output space) the desired output. But if there is little or no vector similarity between the novel and training inputs, or between the desired and training outputs, then the grounds for generalization are weak at best, and performance will likely be poor.

Individual weight-tuning algorithms determine the details of how the appropriate output values can be computed. Classification algorithms generally rely on clustering to learn many-to-one mappings between inputs and outputs. Function approximators can learn many types of mappings that are more nearly one-to-one. This type of learning and generalization is generally quite well-understood. Considerable research is devoted to specifying how generalization is affected by details of architecture, training, smoothness of the mapping, and data sampling (e.g. [6, 43, 48, 70, 77]).

The class of *Standard First-Order Generalization* includes tasks that are readily solved by standard neural networks, such as feedforward back-propagating networks and others that exhibit training independence (See Section 2.1.4). Non-connectionist clustering-based learning algorithms, such as ID3 and C4.5 would also be likely to perform well on such tasks. Problems in this class would typically have a high geometric separability index (i.e. exhibit a high degree of relevant clustering of inputs). The scope of such generalization is limited to novel cases that are within the training space.

Note that the discussion in this section applies to vector similarities that are present in the ground form specification of a task. For example, the data in Table 4.1 shows a pattern of vector similarity that could be useful to a standard neural network (given a larger set of consistent training cases than is shown here). Here, the data would be read as part of a ground form specification, where each vector element (e.g. 0.3) represents the activation value of the corresponding input or output node in the network. Because the clustering (i.e. vector similarity) of the inputs is consistent with the clustering of outputs, a standard back-propagating multilayer perceptron would perform this task well. So might certain biological
networks if, for example, this same ground form task were presented as activation levels in the retina.

However, if a human attempts to solve what appears to be the same problem by reading the digits shown in the table, it becomes clear from introspection that at least some conscious reasoning would be required to conclude that a test output of ‘1’ would be consistent with the training data. In this situation, the data is part of a coarse task specification, because it is a shorthand for sensory and semantic information available to a person. The biological network(s) involved do not necessarily represent the vector elements that a person reads from the table as the activation values of individual nodes.

**Table 4.1: Vector Similarity Task**

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3 0.5 0.5 0.9</td>
<td>1</td>
</tr>
<tr>
<td>0.6 0.1 1.0 0.3</td>
<td>0</td>
</tr>
<tr>
<td>0.7 0.3 0.8 0.5</td>
<td>0</td>
</tr>
<tr>
<td>0.1 0.6 0.4 0.8</td>
<td>1</td>
</tr>
<tr>
<td>0.2 0.7 0.3 1.0</td>
<td>?</td>
</tr>
</tbody>
</table>

This example illustrates how important it is to not to assume that coarse and ground form specifications constitute equivalent tasks even if their data sets look identical when written on the page. Marcus' discussion of the Binary Identity task [52] encounters such a pitfall when the symbol ‘0’ on the page is taken to be comparable to an inactive node (see Sections 3.1.2 and 3.2.1). “Imagining” that one is trained on written data, as Marcus instructs the subjects of his informal observations [53, p. 50], is, of course, not the same as being a network which takes those values as inputs.
CHAPTER 4. REFORMULATION AND CLASSIFICATION

4.3.2 Cross-Dimensional Generalization

The second subclass of First-Order Generalization is Cross-Dimensional Generalization. Tasks in this class are binary classification tasks and can be considered to be learning problems (i.e. they involve weight modification). They would be classified as relational (i.e. Type-2), according to Thornton’s [73, 76] GSI criteria\textsuperscript{12}. However, not all Type-2 problems are included in the class of Cross-Dimensional Generalization. Most of Clark’s and Thornton’s sample tasks [15, 76], particularly those which are presented to humans, would not be included in this class, since they appear to involve considerable conscious reasoning (see Sections 4.2.2 and 4.4.2).

Type-2 tasks that are included in the class of Cross-Dimensional Generalization are binary classification tasks that could be approached by Thornton’s [76] Truth-from-Trash (TFT) strategy (e.g. Figure 4.1). Although Thornton proposes TFT as a general strategy to solve relational (i.e. Type-2) problems, it seems unlikely to perform well on most of the sample relational tasks he describes, since humans appear to be using reasoning to solve them (see Section 4.2.2). Clark’s and Thornton’s discussion of Type-2 generalization focuses mainly on tasks with numeric inputs and a binary scalar output, whereas Cross-Dimensional Generalization includes tasks that require classifying numeric data into more than two distinct classes.

The preceding problem class (Standard First-Order Generalization) depends on vector similarity between neighbouring inputs that is predictive of vector similarity between outputs in the training set. Cross-Dimensional Generalization also depends on vector similarity, but this similarity appears within the task in a somewhat different pattern. Here, vector similarity between entire training cases must be taken into account in order to provide the relevant grounds for generalization.

Understanding this distinction requires clarifying assumptions about what data is available as part of the specification of a generalization task. Previous approaches to related problems

\textsuperscript{12}According to Thornton [73, 76], relational (Type-2) tasks are problematic for back-propagating feedforward networks and boundary-based machine learning algorithms.
(e.g. Type-2 Generalization) are somewhat inconsistent in this regard. Confusion arises, particularly in the treatment by Thornton [76], because of inconsistencies about what is considered to be an ‘input’ to the network. Clarifying this requires distinguishing which of several possible meanings are intended. The term may refer to: the stimulus of the stimulus-response pair that makes up a training or test case; the data that is available to a learning algorithm; or the data presented at the input layer of a neural network. It is important to acknowledge that these are not always equivalent.

In Clark’s and Thornton’s notation and examples [15, 76], the “inputs” are the values on the left-hand-side of their data tables. For example, in Table 2.2 on page 12, the inputs would be the values in the columns labelled \(x_1\) and \(x_2\). This would be consistent with the way that the term “input” is used in Thornton’s definition and application of the Geometric Separability Index [73, 76]. In his characterization of Type-2 learning problems, it is the clustering of such inputs that is relevant. Thornton’s Geometric Separability Index measures the degree to which clustering between these training inputs is predictive of outputs. If such clustering is weak, or if it is not consistent with the classification (i.e. outputs), then the task is considered to be highly relational (Type-2) [15, 73, 76]. According to Thornton, such problems are beyond the scope of feedforward multilayer perceptrons trained by back-propagation, as well as standard (i.e. clustering-based) machine learning techniques [73] (see Section 2.2.2).

However, in Clark’s and Thornton’s sample problems, not just the training inputs, but also the training outputs (e.g. the \(y\) column in Table 2.2 on page 12) are certainly available to humans who are solving the problems by reading the data. The training outputs are also available to back-propagation networks in Clark’s and Thornton’s simulations (i.e. as target values which are used in the training of weights). Further, in Truth-From-Trash (TFT), a general strategy that Thornton outlines in his later work for solving Type-2 problems [76, p. 174], the output data (i.e. class labels) would clearly be available to any algorithms that might instantiate his strategy.

Figure 4.1 shows a sample task which Thornton uses to illustrate his strategy. This task would be a case of Cross-Dimensional Generalization. The horizontal and vertical dimensions of the image correspond to the two dimensions of the original training inputs. Each data point
is labelled by its output value. In the figure, it is clear that neighbouring inputs very commonly belong to different classes. Trying to predict the class labels based merely on the clustering (vector similarity) of the data in the input space would be quite difficult. This can be seen from Figure 4.2, which shows the clustering of the unlabelled training data in the input space.

Figure 4.1: Thornton’s Relational Training Data [76, pp. 171-177]
However, if all of the data that is available as part of such tasks (i.e. inputs as well as outputs of training items) is taken into account, it is possible to discern a new pattern of vector similarity which can support generalization. Cross-Dimensional Generalization identifies a class of problems in which the grounds for generalization depend on vector similarity between training cases. This involves treating the entire training case (i.e. both the input and output) as a single vector.

To illustrate how Cross-Dimensional Generalizations could be justified, consider the data\(^\text{13}\) shown in Table 4.2. Here, the desired test output is ‘1’.

<table>
<thead>
<tr>
<th>Training</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x1$</td>
<td>$x2$</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

\(^{13}\)Although the task Thornton uses to illustrate TFT (Figure 4.1) would be in the class of Cross-Dimensional Generalization, he does not provide complete numerical listings of the training inputs for his example. To support a more precise discussion of the mathematical properties of Cross-Dimensional Generalization, a simplified numerical example is introduced here.
A plot of the input portion of the training cases for this task is shown in Figure 4.3. In this two-dimensional space, two distinct clusters are clearly visible. If the training outputs (i.e. class labels) were consistent with this clustering (e.g. if items labelled ‘1’ were in the cluster on the top left; items labelled ‘0’ were on the lower right), then the task would be straightforward for standard neural networks and clustering-based machine-learning algorithms. The test case (indicated in Figure 4.3 by the symbol ‘?’) could naturally be assigned the label of its nearest cluster. Such a task would be an example of a Standard First-Order Generalization problem.

Figure 4.3: Clustering of Training Inputs

However, the actual task specified in Table 4.2 includes many cases where similar, even neighbouring, inputs are mapped to very dissimilar outputs. This can be seen in Figure 4.4, where each data point is marked by its output value (i.e. class label). The figure illustrates how clustering of the inputs of the training cases is not sufficient to distinguish one class from another.
Yet there is a potentially useful pattern of clustering discernable in the task data. By taking into account the entire training case, including the output values, a number of subclusters can be identified. Figure 4.5 shows a three-dimensional plot of the training cases from Table 4.2, where each case is a vector of the form \((x1, x2, y)\). Here, four distinct (i.e. well-separated) clusters are discernable. These are very consistent with the classification, as can be seen from Figure 4.6, which shows the class label of each training case. Here, each class plane contains two well-separated subclusters.

Identifying such clustering among the training cases can support generalization to novel test items because it reveals groups of uniformly-labelled training cases whose inputs are similar. If the test item is more similar to one of these clusters than the others, then there is grounds for concluding that the test item belongs to same class as that cluster. That is, a class label (i.e. output) can be assigned to a novel test item based on the label of its nearest cluster.

---

14 In Figure 4.6 the symbol ‘*’ marks the centroid of each cluster. The centroids of the four clusters are: \((0.2, 0.8, 1)\), \((0.25, 0.65, 0)\), \((0.8, 0.35, 0)\), \((0.85, 0.15, 1)\).
Figure 4.5: Clustering of Training Cases

Figure 4.6: Subclusters Consistent with Class Labels
Since no class label is included as part of the test case, the comparison must be made on the basis of the inputs of the test case. The nearest cluster to a novel test item can be found by comparing the inputs of the novel test item with the corresponding coordinates of each centroid. One of many comparable measures of the proximity (vector similarity) of two vectors is their inner product. By this measure, the test case (0.4, 0.8, ?) from the sample task in Table 4.2 is nearest to the cluster whose centroid is (0.2, 0.8, 1)\(^{15}\). Classifying the novel test case based its proximity to the cluster centroids of previous test cases thus leads to a novel output of ‘1’.

**Figure 4.7: Centroids of Local Clusters**

![Centroids of Local Clusters](image)

Figure 4.7 shows how the coordinates of the centroids in the input space identify uniformly labelled subclusters. Each asterisk, ‘*’, in the figure represents the location of a centroid. The test case is assigned a label (output) based on the label of its nearest centroid. As described above, the clustering (vector similarity) of entire training cases needs to be taken into account in order to find these subclusters (i.e. the coordinates of their centroids). Thus, generalization is critically dependent on vector similarity between training cases.

\(^{15}\)This is true since the inner product is greater than for the other centroids: (0.4*0.2 + 0.8*0.8) = 0.72 vs. (0.4*0.25 + 0.8*0.65) = 0.62; (0.4*0.8 + 0.8*0.35) = 0.6; (0.4*0.85 + 0.8*0.15) = 0.46.
Cross-Dimensional Generalization extends to novel inputs that lie within the training space, but does not require generalization to novel outputs, since every output value (i.e. class label) has been encountered during training. Note that Cross-Dimensional generalization includes only tasks that have the grounds for generalization described above. That is, problems with test cases whose desired classification is not consistent with the label of their nearest class centroid would not be cases of Cross-Dimensional generalization. The performance of particular generalization techniques (see Chapter 5) depends on how well the details of the architecture, training, and data sampling interact — as is also the case with techniques such as back-propagation for Standard First-Order generalization. Chapter 5 describes how the pattern of vector similarity between training cases described in this section could be exploited by Winner-Take-All networks in ways that appear not to be possible for back-propagating feedforward networks.

4.4 Second-Order Generalization

Second-Order Generalization is distinguished from First-Order Generalization by the ways in which generalizations are justified. Recall that First-Order Generalization depends on patterns of vector similarity present in the training data. Second-Order generalization appears not to be justifiable by such patterns of similarity — either because they are not present, or because they do not support the leap to the novel cases. Sections 4.4.1 and 4.4.2 considers how other types of grounds for generalization are possible.

4.4.1 Core Relational Generalization

In some cases, humans appear to be able to generalize in the absence of the vector similarities that support First-Order generalization. For example, Phillips [63] takes pains to argue that his examples of Cross-Category and Non-Category Generalization extend to novel ‘constituents’ that have little or no similarity to previous training cases, even on the level of their internal semantic representations.
An examination of these and other sample generalization tasks presented in the literature reveals a new pattern of similarity which can support generalization. Generalization in these cases (*Core Relational Generalization* — introduced here) involves two kinds of similarity:

i) similarity (identity) between the output and components of the input *within each case*

ii) *structural similarity between cases.*

Recall, by contrast, that First-Order Generalization depends on vector similarity between elements of *separate* cases. In particular, for Standard First-Order Generalization, vector similarity between inputs of separate training cases is correlated with vector similarity between their corresponding outputs. And *Cross-Dimensional Generalization* depends on vector similarity between training cases.

In *Core Relational* generalization, the training cases have a common pattern of *internal similarity* (as described in point i, above)\(^\text{16}\). When this pattern is consistent over the training set (i.e. the cases are *structurally* similar to each other), it provides the grounds for generalizing to novel test cases that share this internal structure.

Some of the purportedly most radical forms of generalization may be supported in this way. Examples of Core Relational Generalization include interpretations of Phillips' Cross-Category, and Non-Category generalization tasks (Section 2.3.2), as well as Marcus’ Sentence Completion and Binary Identity tasks (Section 2.1). Generalization to radically novel outputs (i.e. those which have little or no similarity to training outputs, or which fall outside of the immediate ‘training’ space) is possible because the structure of the training cases can be abstracted from their particular feature values. Problems in this class are *generation* tasks and, as discussed in Section 4.2.2, learning (i.e. weight change) is likely not the immediate mechanism involved.

---

\(^{16}\)This means that, unlike in *Cross-Dimensional* generalization (which requires vector similarity between training cases), the cases of *Core Relational* generalization need not have any vector similarity to each other whatsoever. That is, vectors representing training and test cases in a *Core Relational* task could all be mutually orthogonal.
Consider the data from Marcus’ Sentence Completion task shown in Table 4.3. The task requires generalization to the test output *blicket*. Here, the output of each training case is identical to the second word of its input. So the training cases are *structurally* similar to each other. Prediction of the novel test output *blicket* can be done on the grounds that the test case would have the same pattern of internal similarity as the training cases. The more consistent the pattern of internal similarity is in the training set, the stronger the grounds for generalization.

**Table 4.3: Sentence Completion Task**

<table>
<thead>
<tr>
<th>Training</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>a rose is a</em></td>
<td><em>rose</em></td>
<td></td>
</tr>
<tr>
<td><em>a lily is a</em></td>
<td><em>lily</em></td>
<td></td>
</tr>
<tr>
<td><em>a tulip is a</em></td>
<td><em>tulip</em></td>
<td></td>
</tr>
<tr>
<td><em>a blicket is a</em></td>
<td><em>?</em></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.8 shows how the training cases share a common internal pattern of similarity (identity) between outputs and elements of the input. The arcs connecting identical words are marked by filled circles.

This pattern of similarity can also be observed in the similarity matrix which compares each training case with itself. Table 4.4 shows the matrix for one case of the Sentence Completion task. Each element, $S_{ij}$, of the matrix is equal to 1 when word $i$ is identical to word $j$; otherwise, $S_{ij} = 0$. The matrix is symmetrical, with diagonal entries of 1.
Figure 4.8: Structure of the Sentence Completion Cases

Table 4.4: Similarity Matrix

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>rose</th>
<th>is</th>
<th>a</th>
<th>rose</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rose</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rose</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
The off-diagonal elements in Table 4.4 correspond to the pattern of internal similarity shown in Figure 4.8. For readability, Figure 4.9 shows this pattern in a simplified graphical notation which can be compared with the off-diagonal entries shown in Table 4.5.

Figure 4.9: Internal Structure as Off-Diagonal Matrix Elements

Table 4.5: Off-Diagonal Elements of the Similarity Matrix
Similarly, the structure of Phillips’ Chair-Colour task is shown in Figure 4.10 and Table 4.6.

**Figure 4.10: Internal Structure of Phillips’ Chair-Colour Task**

**Table 4.6: Similarity Matrix for the Chair-Colour Task**
Although only shaded regions\(^{17}\) of the similarity matrices shown in Table 4.5 and Table 4.6 are needed to provide grounds for generalization in these two examples, the entire matrix is considered here, because it is relevant to the more general method of computing the \textit{Relational Trace} of a task.

The \textit{Relational Trace} of a task is an array whose elements indicate the pattern of internal similarity that is relatively stable over the training set. As can be seen from the above discussion, for the Sentence Completion and Chair-Colour tasks, the pattern of internal similarity is very consistent across the training cases. In the Sentence Completion task, for every training and test case, the output is always identical to the second word of the input. In the Chair-colour task, for every training and test case, the output is always identical to the fifth word of the input. Even more simply, in the Binary Identity task in Table 3.1, the output is always identical to the input. But if any of the cases had been inconsistent with the structure of the others in the same task, the grounds for generalization would have been weakened. So it is important to take into account how consistently the pattern appears within the training cases. Chapter 5 explores how this might be accomplished.

\subsection*{4.4.2 Composite and Special-Purpose Relations}

This class includes problems that appear not to be directly addressed by any single technique that would work for the classes discussed so far in this chapter. Some of these tasks appear to be related to previous problem classes in that they have similar grounds for generalization. For example, the information in the two-dimensional relational trace may cast light on the tasks encountered in the studies on Infant Habituation [56] and Hebrew Word Formation [8], as well as the relational similarity example (Figure 2.2 on page 18) of Goldstone et al. [28]. These problems appear to be related to the class of Core Relational Generalization, in that they depend on patterns of similarity (identity) \textit{within} the training cases, which is compared with the structure of the test case. This suggests that techniques which involve computing the two-

\(^{17}\)These correspond to the right-most diagonals in Figure 4.9 and Figure 4.10, respectively. These indicate which element of the input of each training case is predictive of the output.
dimensional relational trace — perhaps as a module within a larger network — may be relevant. Since the relational trace can be computed using node activations, rather than edge weight modifications, it suggests way to understand such tasks in which learning may not be the immediate mechanism.

For example, Figure 4.11 illustrates the pattern of internal similarity that is crucial in the grounds for distinguishing the infants’ habituation sentences from test sentences. Table 4.7 shows how this would be reflected in the similarity matrices for these three sentence forms. Suppose a two-dimensional (triangular) array of three trace modules (see Section 5.2.1) were used to extract the pattern of similarity that is stable over the habituation sentences. The relational trace, in this sense, would be a kind of prediction about the expected structure of the next sentence. This would provide information that would be central to determining whether radically novel test cases are consistent or inconsistent with habituation. That is, if the test case produces a sudden change in the relational trace, then the sentence is inconsistent with habituation; otherwise, it is consistent.

Figure 4.11: Structure of Habituation Sentences
PART of the difficulty in classifying such tasks is that they appear to have a somewhat different form than most of the sample tasks described in the literature. These tasks differ from Core Relational Generalization in the sense that they are not clearly generation tasks. Further, unlike many typical generalization tasks, the behavioural responses observed in the studies do not correspond in an obvious way to particular output vectors that might be generated by a neural network. The response in the infant habituation study is a change in duration of attention. In the word formation study, it is a similarity judgement, as measured by rating scales and response times. The experiment by Goldstone et al. required subjects to make a judgement of relative similarity.

Consider the relational similarity example (Figure 2.2 on page 18) of Goldstone et al. [28]. The two types of similarity that the researchers observe (i.e. "physical" vs. "relational") can be seen to involve making two different types of comparisons. Table 4.8 a) shows how a judgement based on physical similarity might be understood. Here, (O, O) can be seen to be more physically similar to (O, □) than to (□, □), since the similarity matrix that compares (O, O) with (O, □) shows some partial similarity between the two cases, whereas the matrix that compares (O, O) with (□, □) shows none. Table 4.8 b) shows how a relational similarity comparison might be supported. Here, the internal similarity matrix for the case (O, O) is
identical to the one for (□, □). But the internal similarity matrix for (O, □) is different from either of these.

According to the researchers, the ability to make judgements based on relational similarity appears to be the later-developing skill, [28]. The analysis illustrated in Table 4.8 shows how the grounds for these two different types of “generalization” involve different ways of using the similarity matrix. In the physical similarity judgement, two similarity matrices must be compared. In relational similarity, three must be compared. That is, the pattern of internal similarity of the sample case (O, O) must be extracted and compared with that for the two ‘test’ cases, (O, □) and (□, □).

Table 4.8: Structure of Relational Similarity Task

<table>
<thead>
<tr>
<th></th>
<th>O</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>□</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>□</th>
<th>□</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

a) physical similarity

<table>
<thead>
<tr>
<th></th>
<th>O</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>O</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>□</th>
<th>□</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

b) relational similarity
Another group of tasks which may be related to Core Relational Generalization can be described as linking problems. Although previous formulations of the research problem typically consider relations to play an essential role in distinguishing classes of learning problems, identity is the key relation in a surprising majority of their sample tasks. Some, such as string reversal and concatenation, involve more complex or composite structural patterns than do tasks in the class of Core Relational generalization but, even in such cases, identity is still the key relation that links elements of inputs to elements of outputs.

In linking tasks, part of the novel output to be generated for the test case is identical to some part of the input for each case. So tasks in the class of Core Relational generalization are one form of linking tasks. In a second form of linking task, the output to be generated is not the final word of the sentence. For example$^{18}$, consider the task:

Training cases:  

<table>
<thead>
<tr>
<th>John washed 6 dogs. 6 dogs did John wash.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jane herded 11 cats. 11 cats did Jane herd.</td>
</tr>
<tr>
<td>Bill fed 20 fish. 20 fish did Bill feed.</td>
</tr>
</tbody>
</table>

Test case:  

| Mary saw 4 birds. ___ birds did Mary see. |

Here, the desired response (i.e. 4) in the test case is identical to a preceding part of the test case. Since the remaining words are given (i.e. they need not be predicted), the problem is somewhat unusual in form for a generalization task. One way to break the problem into inputs and outputs is as is shown in Table 4.9.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td></td>
</tr>
<tr>
<td>John washed 6 dogs</td>
<td>6</td>
</tr>
<tr>
<td>dogs did John wash.</td>
<td>11</td>
</tr>
<tr>
<td>cats did Jane herd.</td>
<td>20</td>
</tr>
<tr>
<td>Test</td>
<td></td>
</tr>
<tr>
<td>fish did Bill feed.</td>
<td>?</td>
</tr>
</tbody>
</table>

18I am grateful to my supervisor, Dr. Robert F. Hadley, for suggesting this type of example.
Viewed in this way, the task has the same form as key Core Relational generalization tasks: Marcus' Sentence Completion, as well as Phillips' Cross-Category and Non-Category examples, and so could be approached using the Relational Trace network described in Section 5.2. Section 5.2.2 describes how tasks with this form would be handled by the Relational Trace network. However, viewing this linking task in the way shown in Table 4.9 would depend on knowing in advance which word is the one which is to be predicted. As Figure 4.12 illustrates, there are other 'linked' words that could be predicted based on the information in this more complete relational trace (i.e. the off-diagonal elements of the internal similarity matrix) for the task. So a network that could be ready to predict words in any of these three positions might include components (e.g. an array of trace modules) to compute the more complete relational trace of the task.

**Figure 4.12: Internal Structure of a Task with Multiple Links**

- John washed 6 dogs 6 dogs did John wash.
- Bill fed 20 fish 20 fish did Bill feed.
- Jane herded 11 cats 11 cats did Jane herd.
- Mary saw 4 birds 4 birds did Mary see.
Alternatively, if the test case required generating the entire final sentence, \textit{4 birds did Mary see}, when given the input, \textit{Mary saw 4 birds.}, then the task would be related\textsuperscript{19} to a different type of linking problem, as described below. In this third type of linking problem, the grounds for generalization involve a somewhat more complex pattern of similarity. As is true in general for linking tasks, parts of the novel output to be generated for the test case is identical to some part of the input of that case. However, in this third type of linking problem, the rest of that test output is identical to part of the previous training outputs. So, for example, in the output, \textit{Mary saw 4 birds}, predicting the word \textit{did} requires observing its occurrence in previous training cases, rather than within the current test case (see Figure 4.12).

Other examples of this third type of linking problem include Marcus’ conception of concatenation as a UQOTOM (see Section 4.1.1) and Hadley’s rapid induction example. Hadley [35] states that “English-speaking adults have no difficulty inducing a novel pattern, and completing the final ‘sentence’ in the following series: ‘Rose biffle biffle zarple zarple rose; Frog biffle biffle zarple zarple frog; Blicket biffle biffle \textit{____ _____}’.” Here, zarple zarple Blicket would be a justifiable generalization, on the grounds that the final word of the sentence corresponds to the first, consistently throughout the training set. But the fourth and fifth words could only be generated on the grounds that they are identical to the fourth and fifth words of the previous sentences. (Note that this means that generalizations with this justification do not include those with novel items in the fourth and fifth words of the sentences — these words must always be zarple). Considering that the information in a relational trace would provide justification for the novel part of the output, an array of trace modules might be useful as part of a network that could solve such linking problems.

Marcus does briefly mention one example (see Section 4.1.1) of the kinds of relations (other than linking or identity) that he has in mind in his definition of Universal Generalization. He states that multiplication would be an example of a UQOTOM that would be consistent with his definition of Universal Generalization [53, p. 45], but does not describe the type of

\textsuperscript{19}Technically, this particular task is not simply a linking problem, since it also requires the ability to conjugate irregular verbs.
task he has in mind. It is difficult to guess how multiplication might be cast as a Universal Generalization problem. For most adults, performing multiplication would seem to depend on a previously-learned procedural skill or a set of memorized associations, rather than on learning as part of the immediate task. Even if multiplication is being encountered for the first time, would observing a handful of examples of multiplication by 4 (e.g. 7 -> 28, 5 -> 20, 9 -> 36) really allow one to generalize to a novel case (e.g. 6 -> ?) ? In what sense is does 6 fall outside the training space?

Clark and Thornton also consider various types of relations. But in most of their sample tasks, conscious serial reasoning, rather than learning, appears to be involved. For example, humans who read the task shown in Table 2.2 on page 12, are unlikely to be able to solve it if they haven’t previously learned how to calculate the difference between two numbers. Tasks such as n-bit parity [15] may involve previously learned procedural skills, such as counting. Some such tasks may be considered to include components or aspects of previous classes, to the extent that they depend on previous learning of the special-purpose mappings (e.g. memorizing multiplication tables, by First-Order methods).

Finally, tasks remain which seem to be intimately linked with learning, due to their connection with early language acquisition. Examples include learning to form the past or progressive tense and plurals of words [53] and some recent interpretations of the systematicity problem [32, 60]. Depending on the training data that is available, learning the tenses of regular verbs and the plurals of regular nouns may be seen to have the structure of linking problems. Compared to problem classes described earlier in this chapter, some of these tasks appear to involve even more complex patterns of relations. Cases of linguistic systematicity which require learning relatively simple grammars may involve multiple relations between subjects, verbs, objects, multiple levels of embedding, and even semantics. For example, Hadley and Hayward [38] propose a challenging form of systematicity — Strong Semantic Systematicity (see Section 2.3.1) — which requires training a network to determine the meanings of sentences containing words in novel syntactic positions and at novel levels of embedding. They present a Hebbian-based method which uses a form of binding to perform a sample task of
this type. During training, sentences from the following recursive grammar are presented one at a time.

\[
S \rightarrow NP \lor NP \\
NP \rightarrow N \lor N \, RC \\
N \rightarrow \text{Mary} \lor \text{Jane} \lor \text{Sally} \lor \text{Susan} \lor \text{Vicky} \lor \text{Fran} \lor \text{Abe} \lor \text{Bill} \lor \text{Carl} \lor \text{Dave} \lor \text{Earl} \lor \text{Fred} \\
V \rightarrow \text{likes} \lor \text{knows} \lor \text{treats} \lor \text{calls} \lor \text{draws} \lor \text{helps} \lor \text{races} \lor \text{sees} \\
\text{REL-PRO} \rightarrow \text{who} \\
\text{RC} \rightarrow \text{REL-PRO} \land V \lor NP
\]

The meaning of each sentence is also presented during training, by activating nodes of a semantic network to represent a proposition. From this, the network must learn to associate the sentences of the grammar with their corresponding meanings and generalize this ability to sentences that include words in novel syntactic positions and at novel levels of embedding.

### 4.5 Summary

This chapter has explored some of the ways in which previous formulations of the research problem may be seen in relation to each other. The two main problem formulations — Marcus' *Universal Generalization* [52, 53] and Clark’s and Thornton’s *Type-2 Generalization* [15] — both claim to describe problems that are beyond the scope of multilayer perceptrons. But the formulations differ in the ways in which test cases can be novel (outputs that exceed the training space vs. previously unseen inputs); and because each applies to tasks of a distinct form (generation vs. classification), they can be considered to describe two distinct classes of problems. While *Cross-Category* and *Non-Category* generalization may be seen to be cases of Universal Generalization [63], suggestions that certain versions of *Systematicity* are comparable to Universal Generalization are more controversial [21, 32, 35, 60]. Differing viewpoints can be seen to reflect differing assumptions about representation, novelty, and terminology. Finally, studies of *Relational Similarity* [28] in humans suggest that relations play an important
role, not just in theoretical distinctions between types of generalization[15, 52, 53], but in observable differences between ways in which humans can generalize.

This thesis identifies two key factors that can be used to help integrate the insights of previous researchers into a more unified understanding of the problem space: grounds for generalization (based on how distinct types of similarity appear within the task) and the role of learning. Based on these two underlying factors, a distinction is drawn between two distinct and mutually exclusive classes: First-Order Generalization and Second-Order Generalization.

First-Order Generalization (which consists of the two subclasses: Standard First-Order and Cross-Dimensional Generalization) includes tasks that can be considered to be learning problems. Such generalizations are made on the grounds that there is useful vector similarity between inputs and outputs (for Standard First-Order tasks) or between entire training cases (for Cross-Dimensional Generalization). Standard First-Order tasks are those which have been considered to be relatively unproblematic for standard connectionist and symbolic learning techniques. Cross-Dimensional Generalization problems would be classified as Type-2 according to GSI criteria presented by Thornton [73, 75].

Second-Order Generalization consists of the two subclasses: Core Relational Generalization and generalization involving Composite or Special-Purpose Relations. In Second-Order generalization, vector similarity does not provide sufficient grounds for generalization. Cross-Dimensional generalizations are made on the grounds that the output is consistently identical to some element of the input within each sample ‘training’ case. Learning (i.e. weight change) is not considered to be the immediate mechanism in Cross-Dimensional Generalization, due in part to the time scales involved (see Section 4.2.2). This chapter introduces the Relational Trace, an array that encodes the pattern of similarity with training cases which, when consistently present in the training set, provides the grounds for Cross-Dimensional Generalization. Such problems can be approached by the architecture proposed in Chapter 5 (Section 5.2), which computes the relational trace.

Finally, tasks that involve Composite or Special-Purpose Relations appear not to be directly addressed by any single technique that would succeed for the first three subclasses. This
subclass includes the tasks from the infant study [55] and Hebrew word formation study [8], the relational similarity task [28], and various linking problems. For these first tasks, the general method of computing the relational trace of the task can be seen to provide information that would be key in their solution. This suggests that possible solutions might incorporate arrays of trace modules from the architecture presented in Chapter 5 (Section 5.2.1). Also included in this final subclass class are those Type-2 problems in which vector similarity does not provide grounds for generalization and Systematicity, which may involve multiple complex relations and has associations [36] with early language acquisition.

The subclasses and properties of the two main classes of generalization are summarized in Table 4.10. Ways in which Winner-Take-All networks could be used in the solution of Cross-Dimensional Generalization and Core Relational Generalization tasks are explored in the following chapter.

**Table 4.10: Classes of Generalization Tasks**

<table>
<thead>
<tr>
<th>Subclass</th>
<th>Grounds (Similarity)</th>
<th>Learning</th>
<th>Scope (Novelty)</th>
<th>Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>First Order</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard First-Order</td>
<td><strong>Vector</strong>: between inputs; between outputs</td>
<td>Weight-Change</td>
<td>Within the Training Space</td>
<td>Multilayer Perceptrons</td>
</tr>
<tr>
<td>Cross-Dimensional</td>
<td><strong>Vector</strong>: between training cases</td>
<td>Weight-Change</td>
<td>Across nodes</td>
<td>Winner-Take-All&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td><strong>Second Order</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core Relational</td>
<td><strong>Identity</strong>: between outputs and portions of inputs; <strong>Structural</strong>: between cases</td>
<td>Post-Learning</td>
<td>Radically Novel (beyond training space of immediate sample set)</td>
<td>Relational Trace&lt;sup&gt;b&lt;/sup&gt; (WTA components)</td>
</tr>
<tr>
<td>Composite/Special-Purpose</td>
<td>Various combinations of previous types of learning and generalization. See Section 4.4.2 for details.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

<sup>a</sup> As described in Chapter 5, this class (Cross-Dimensional Generalization) is problematic for feedforward multilayer perceptrons trained by back-propagation.

<sup>b</sup> The relational trace technique and architecture, introduced in Chapter 5, are among the novel contributions of this thesis.
Chapter 5

Techniques

This chapter explores how Winner-Take-All networks can be used to approach classes of problems identified in Chapter 4. In doing so, it addresses the final component of the thesis problem statement:

v) How might unsupervised connectionist techniques help to address such challenges?

Section 5.1 explores how Cross-Dimensional Generalization problems could be addressed by Winner-Take-All networks. Section 5.2 shows how Winner-Take-All networks could play a role in the solution of Core Relational Generalization problems. The chapter concludes with a summary in Section 5.3.

5.1 Winner-Take-All Networks

Figure 5.1 shows the architecture of a simple Winner-Take-All (WTA) network. Here, the input layer is fully connected to the competitive layer by excitatory (i.e. positively weighted) feedforward connections. A competitive layer consists of one or more competitive clusters of nodes. The competitive layer in Figure 5.1 consists of a single competitive cluster of four nodes. Nodes within a cluster are fully connected to each other by mutually inhibitory (i.e. negatively weighted) connections.
Nodes in a competitive cluster are called Winner-Take-All nodes. The output $b_k$ of a Winner-Take-All node satisfies:

$$b_k = \begin{cases} 1 & \text{if } v_k > v_j \quad \forall j, j \neq k \\ 0 & \text{otherwise} \end{cases}$$

(5.1)

where $v_k$ is the combined action of all its forward and feedback (lateral) inputs [42, p. 59]. That is, for a given input pattern, a given Winner-Take-All node wins (i.e. is activated) if and only if the sum of its weighted inputs is greater than those of all other nodes in that competitive cluster. So for each input pattern, only a single node in the competitive cluster will be activated. For a cluster with equal inhibitory weights, the active node will be the one whose weights from the input layer most closely approximate that input vector.

Winner-Take-All networks are trained by an unsupervised process of Competitive Learning. The standard competitive learning weight-update rule\(^1\) satisfies:

$$\Delta w_{kj} = \begin{cases} \eta (x_j - w_{kj}) & \text{if neuron } k \text{ wins} \\ 0 & \text{if neuron } k \text{ loses} \end{cases}$$

(5.2)

where

---

\(^1\) This rule is also known as the Kohonen learning rule [10, 45, 17, 42, p. 59].
CHAPTER 5. TECHNIQUES

\( \eta \) is the learning rate,
\( x_j \) is the activation of node \( j \),
\( w_{kj} \) is the weight from input node \( j \) to node \( k \), and
\( \Delta w_{kj} \) is the modification to \( w_{kj} \).

When neuron \( k \) wins, a proportion of the weight from its inactive inputs is shifted to its active inputs. The result is that the weight vector \( \vec{w}_k \) is moved slightly toward the input pattern \( \vec{x} \). If there is clustering present in the data, output nodes will compete with each other to represent the clusters. Stability of the network depends on whether the clusters are "sufficiently distinct” [42, p. 59]. After training, the input weight vector of a WTA node corresponds to the centroid of the cluster it represents. A test input will activate the WTA node whose cluster centroid is nearest to that input vector. That is, the winning WTA node is the one whose vector of incoming weights is most similar to the test vector.

5.1.1 Techniques for Cross-Dimensional Generalization

As discussed in Section 4.3.2, in Cross-Dimensional generalization tasks, the clustering among the inputs of the training cases is not sufficient to predict the outputs. However, clustering among training cases can provide grounds for generalization in such tasks. Cross-dimensional generalization extends to novel outputs whose classification is consistent with the clustering of the training cases.

Since, in the cases under consideration, both the inputs and outputs of training cases are available to humans, back-propagating networks, and algorithms that would instantiate Thornton’s Truth-From-Trash strategy, it is reasonable to expect that they would also be available when such tasks are presented to unsupervised neural networks, such as Winner-Take-All networks. Unlike back-propagation networks, WTA networks do not make use of training outputs as target values for the output layer. In order to make outputs of the training cases available to a WTA network, they would need to be included as part of the vector that is presented at the input layer.
So the most directly comparable tasks that could be presented to a WTA network would be ones in which the entire training case is presented at the input layer as a single vector (i.e. the concatenation of the training input vector with the training output vector). Including the training output as part of the vector that is presented at the input layer means that the pattern of clustering between training cases that can support Cross-Dimensional generalization is present in this expanded input space. This section shows how such a pattern of clustering can be exploited by Winner-Take-All techniques to perform Cross-Dimensional Generalization.

To illustrate this, the cross-dimensional task that was introduced in Section 4.3.2 (Table 4.2) is revisited here. Table 5.1 a) shows the data for the original task. Two comparable ground form tasks which could be presented to a WTA network are shown in shown in b) and c) of Table 5.1. The original task a) required generalization to the novel test output of ‘1’. In the corresponding tasks shown on the right of Table 5.1, the correct test output would be ‘1’ for task b) and (1, 0) for task c). In both of these ground form specifications, the outputs of the training cases (i.e. the class labels) are made available to the WTA network as part of the input vector, rather than used as target values for the outputs during training.

Task b) requires the most limited recoding of the data and is the simplest to visualize, since its input space is only three-dimensional. Task c) uses an alternative representational scheme — one which is used later in this section to show how the technique could be extended to tasks that classify inputs into more than two classes.

In task b), the class label is included as a third input, $x_3$, so that the (expanded) input vector presented at the input layer of the network is $(x_1, x_2, x_3)$. Here, $x_3 = -1$ (where the initial class label was 0) so that novel test cases can be presented without class labels ($x_3 = 0$). So, for example, the first training case in task b) would be presented to the network as the (expanded) input vector $(0.8, 0.1, 1)$. The third training case would be $(0.3, 0.6, -1)$ and the test case would be presented as $(0.4, 0.8, 0)$. For test cases, an unknown class label is indicated by a value of $x_3 = 0$, since this value provides no activation to the input label node. A value of zero is neutral (i.e. equally distant) relative to the values, +1 and -1, which are used to represent the class labels in this task.
### Table 5.1: Comparable Cross-Dimensional Generalization Tasks

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Input</th>
<th>Output</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$y$</td>
<td></td>
<td>$x_1$</td>
<td>$x_2$</td>
</tr>
<tr>
<td>0.8</td>
<td>0.1</td>
<td>1</td>
<td></td>
<td>0.8</td>
<td>0.1</td>
</tr>
<tr>
<td>0.3</td>
<td>0.9</td>
<td>1</td>
<td></td>
<td>0.3</td>
<td>0.9</td>
</tr>
<tr>
<td>0.3</td>
<td>0.6</td>
<td>0</td>
<td></td>
<td>0.3</td>
<td>0.6</td>
</tr>
<tr>
<td>0.9</td>
<td>0.2</td>
<td>1</td>
<td></td>
<td>0.9</td>
<td>0.2</td>
</tr>
<tr>
<td>0.7</td>
<td>0.4</td>
<td>0</td>
<td></td>
<td>0.7</td>
<td>0.4</td>
</tr>
<tr>
<td>0.9</td>
<td>0.3</td>
<td>0</td>
<td></td>
<td>0.9</td>
<td>0.3</td>
</tr>
<tr>
<td>0.2</td>
<td>0.7</td>
<td>0</td>
<td></td>
<td>0.2</td>
<td>0.7</td>
</tr>
<tr>
<td>0.1</td>
<td>0.7</td>
<td>1</td>
<td></td>
<td>0.1</td>
<td>0.7</td>
</tr>
<tr>
<td>0.4</td>
<td>0.8</td>
<td>?</td>
<td></td>
<td>0.4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

In Table 5.1 c), the class labels (i.e. training outputs) are included as part of the (expanded) input vector, $(x_1, x_2, x_3, x_4)$, which is presented to the input layer of the network. That is, the inputs, $x_3$ and $x_4$, correspond to the outputs $y_1$ and $y_2$. Here, $(1, 0)$ indicates a class label of 1, and $(0, 1)$ indicates a class label of 0. In the novel test case, no class label is presented to the network, as indicated by the values $(0, 0)$ in the label portion of the test input vector. So test cases do not activate the label nodes, and the values $(0, 0)$ are neutral (i.e. equally distant) relative to the values representing the two classes: $(0,1)$ and $(1, 0)$. 
The two ground form tasks, b) and c) from Table 5.1 could be approached by a WTA network with two competitive layers. Figure 5.2 shows a network that could be used for Task c). The input layer consists of \( l = 2 \) regular input nodes, as well as additional input nodes for the class labels, where \( m \) is the number of distinct output class labels in the training set. For tasks b) and c), which are binary classification problems, \( m = 2 \). The first competitive layer of the network consists of \( n = 4 \) nodes which are fully connected to each other by mutually inhibitory links, as is standard in a competitive layer. In general, if there are \( p \geq 2 \) distinct subclusters in each of the \( m \) classes, network requires a hidden WTA layer of size \( n = m^2 p \). The second competitive layer (which is also the output layer of the network) consists of \( m = 2 \) mutually inhibiting nodes. Each layer is fully connected to its neighbouring layers. The output layer is also fully connected to the two input nodes where the class labels are presented. The entire network can be thought of as a cascade of two single-layer WTA networks.

The network would be trained by competitive learning. During training, competitive learning moves the weights entering each of the four nodes in the first competitive layer to converge to the centroids of each of the four main local clusters. This could be achieved, for

---

To handle the task specified in Table 5.1 b), the architecture would be the same, except that one of the Class Label input nodes would be omitted. The output of either node in the WTA(2) layer could be used to discriminate between the two classes. The activation of one would be consistent with the labelling. The values of the other would be its converse.
example, by the version of competitive learning presented by Rumelhart and Zipser [67]. They show that, if there are \( n \) WTA units in a competitive layer, then the inputs to that layer will be divided into \( n \) clusters [67, 42, pp. 58-60]. This will be stable as long as the clusters are "sufficiently distinct" [42, p. 60]. So, for example, if clusters overlap, the network will be unstable. However, if the distance between training items in different clusters is somewhat greater than the distance between training items within each cluster, the network will converge to a stable state.

As discussed in Chapter 4 (see Section 4.3.2), Cross-Dimensional Generalization tasks exhibit such clustering among training cases. That is, the clusters in the expanded input space are well-separated, even though the formal inputs (i.e. \( x_1 \) and \( x_2 \)) in the original input space (see Figure 5.3) were not geometrically separable. Figure 5.3 shows how clustering of the (formal) training inputs in the original input space would suggest a different classification than the one indicated by the class labels of the training cases (see Figure 5.4).

Figure 5.3: Clustering of Training Inputs in the Original Input Space

---

3 In addition, Rumelhart's and Zipser's model [67] assumes that vectors are normalized in length and convergence depends on training cases being presented in random order with equal frequency.
However, by including the class labels as part of each input vector, training cases from distinct classes are mapped to separate planes in the expanded input space (see Section 4.3.2). Figure 5.5 illustrates how this is the case for task b), where the expanded input space is three-dimensional. By including the class labels as part of the training inputs to the network, each training case (i.e. its expanded vector) belongs to a separate plane of the expanded input space. That is, training cases from the class with the input label $x_3 = 1$ are contained in the plane that is orthogonal to the $x_3$-axis at $x_3 = 1$. Likewise, the plane that is orthogonal to the $x_3$-axis at $x_3 = -1$ contains the cases whose label is $x_3 = -1$. For example, the first training case, $(0.8, 0.1, 1)$, would be in the plane of $x_3 = 1$. The test case, $(0.4, 0.8, 0)$, would be equidistant from the two planes, since $x_3 = 0$.

Figure 5.4: Classification to be Learned

---

4 In Figure 5.5 the symbol ‘*’ marks the centroid of each cluster. The centroids of the four clusters are: $(0.2, 0.8, 1)$, $(0.25, 0.65, 0)$, $(0.8, 0.35, 0)$, $(0.85, 0.15, 1)$. 
Binary Classification

For task b) and others of its form, it can be shown that training items from the same class will always be closer to each other in the expanded input space than training items from different classes. To begin with, let us consider tasks with formal input vectors with \( l = 2 \) elements, where those elements have values in the closed interval \([0, 1]\), and where input class labels are indicated by values of \( \pm 1 \). The following discussion begins by focusing particularly on tasks of this form to show how a WTA network can perform Cross-Dimensional Generalization — since the expanded input space for tasks of this restricted form is three-dimensional, it is more easily visualized than the higher-dimensional spaces involved with tasks of the more general form (i.e. with \( l \geq 2 \) inputs whose values are in the closed interval \([0, R]\), and with input label values of \( \pm \Omega \)). Results for the more general case are introduced during

---

5 By definition, the closed interval \([u, v]\) = \( \{x \in \mathbb{R} | u \leq x \leq v\} \) [58, pp. xv-xvii].
the course of this discussion, with details provided in the Appendix. Ways in which the WTA technique could be extended to tasks that classify inputs into \( m \geq 2 \) distinct classes are presented in the following section. In some tasks of the more general form, the class planes may not be sufficiently separated in the expanded input space (depending on the relative values of \( I, R, \) and \( Q \)). However, such tasks could be handled by representing the input class labels by a larger value of \( Q \), according to equations described later in this section and derived in the Appendix (Sections A.1 and A.2).

For the restricted case described above, we can say without qualification that, within the expanded input space, training cases from the same class are more similar to each other than are training cases from separate classes. As can be seen from Figure 5.5, the distance between any two items in the same class will be at most \( \sqrt{2} \), since they share a common class plane. But since any two items from different classes will belong to different planes, they will be separated by a distance of at least 2.

It is noteworthy, however, that in implementations of competitive learning that use normalized inputs, such as Rumelhart's and Zipser's method [67], the angle between vectors is a more reliable measure of similarity than is Euclidean distance\(^6\). This is because normalization can change the Euclidean distance between a pair of vectors but not the angle between them (since normalization changes the magnitude but not the direction of a vector).

Further, the angle between vectors is the more relevant measure for most implementations of competitive learning, since activation values of WTA nodes are typically computed as the inner product of the input vector with its incoming weight vector. The relationship between the inner product of two vectors, \( \mathbf{x} \) and \( \mathbf{y} \), and the angle, \( \theta \), between them is \( \mathbf{x} \cdot \mathbf{y} = |\mathbf{x}| \cdot |\mathbf{y}| \cos \theta \). It holds for any two vectors in \( \mathbb{R}^n \) [58, p. 48].

Let \( \mathbf{a} = (a_1, a_2, \ldots, a_I) \neq \mathbf{b} = (b_1, b_2, \ldots, b_I) \) be any two distinct training vectors from the original input space. Then, their corresponding inputs in the expanded input space will be of the form: \( \tilde{\mathbf{a}} = (a_1, a_2, \ldots, a_I, a_{I+1}) \), \( \tilde{\mathbf{b}} = (b_1, b_2, \ldots, b_I, b_{I+1}) \), where, \( \forall \ i \leq l, a_i, b_i \in [0, 1] \)\(^7\), and \( a_{I+1} \)

---

\(^6\) Some competitive learning methods avoid the need for normalizing inputs by using Euclidean distance measures [17, 18], rather than inner product similarity measures. The relevant proofs for these are given in the Appendix (Sections A.1.2 and A.2.2).
, \( b_{l+1} \in \{-1, 1\} \) are the class labels presented at the input layer. Consider first, the restricted case, where \( l = 2 \).

If \( \mathbf{\tilde{a}} \neq \mathbf{\tilde{b}} \) are from the same class, then the angle \( \alpha \) between the expanded vectors, \( \mathbf{\tilde{a}'} \) and \( \mathbf{\tilde{b}'} \), will be the greatest for pairs of vectors at diagonally opposed corners of the plane (see Figure 5.5). There are two possibilities:

i) \( \mathbf{\tilde{a}'} = (1,1,1) \) and \( \mathbf{\tilde{b}'} = (0,0,1) \)

ii) If \( \mathbf{\tilde{a}'} = (1,0,1) \) and \( \mathbf{\tilde{b}'} = (0,1,1) \).

In case i), \( \cos \alpha = (\mathbf{\tilde{a}' \cdot \tilde{b}'}) / (|\mathbf{\tilde{a}'| \cdot |\tilde{b}'|}) = (1,1,1) \cdot (0,0,1) / \sqrt{3} = 1 / \sqrt{3} \), so \( \alpha = \cos^{-1}(1 / \sqrt{3}) = 54.7^\circ \).

In case ii) \( \cos \alpha = (\mathbf{\tilde{a}' \cdot \tilde{b}'}) / (|\mathbf{\tilde{a}'| \cdot |\tilde{b}'|}) = (1,0,1) \cdot (0,1,1) / 2 = 1 / 2 \), so \( \alpha = \cos^{-1}(1 / 2) = 60^\circ \).

So, for training cases from the same class, \( \alpha \leq 60^\circ \).

Likewise, let \( \mathbf{\tilde{c}} = (c_1, c_2, ..., c_l) \neq \mathbf{\tilde{d}} = (d_1, d_2, ..., d_l) \) be any two distinct training vectors from the original input space. Then, their corresponding inputs in the expanded input space will be of the form: \( \mathbf{\tilde{c}'} = (c_1, c_2, ..., c_l, c_{l+1}) \), \( \mathbf{\tilde{d}'} = (d_1, d_2, ..., d_l, d_{l+1}) \).

If \( \mathbf{\tilde{c}} \neq \mathbf{\tilde{d}} \) are from different classes, then the angle \( \beta \) between the expanded vectors, \( \mathbf{\tilde{c}'} \) and \( \mathbf{\tilde{d}'} \), will be the least for vectors at the corners of a their respective class planes, where \( \mathbf{\tilde{c}} \approx \mathbf{\tilde{d}} \). Since \( \mathbf{\tilde{c}} \neq \mathbf{\tilde{d}} \), \( \beta \) will be at least as large as the minimum of the following:

i) \( \mathbf{\tilde{c}'} = (0,0,1) \) and \( \mathbf{\tilde{d}'} = (0,0,-1) \)

ii) \( \mathbf{\tilde{c}'} = (1,0,1) \) and \( \mathbf{\tilde{d}'} = (1,0,-1) \).

iii) \( \mathbf{\tilde{c}'} = (1,1,1) \) and \( \mathbf{\tilde{d}'} = (1,1,-1) \).

In case i), \( \cos \beta = (\mathbf{\tilde{c}' \cdot \tilde{d}'}) / (|\mathbf{\tilde{c}'| \cdot |\tilde{d}'|}) = (0,0,1) \cdot (0,0,-1) / 1 = -1 \), so \( \beta = \cos^{-1}(-1) = 180^\circ \).

In case ii), \( \cos \beta = (\mathbf{\tilde{c}' \cdot \tilde{d}'}) / (|\mathbf{\tilde{c}'| \cdot |\tilde{d}'|}) = (1,0,1) \cdot (1,0,-1) / 2 = 0 \), so \( \beta = \cos^{-1}(0) = 90^\circ \).

---

7 That is, \( a_i \) and \( b_i \) are in the closed interval \( [0, 1] \), which means that \( 0 \leq a_i \leq 1 \) and \( 0 \leq b_i \leq 1 \). Here, braces, \{\}, are used to enclose a list of elements, as in standard set notation [49, pp. 5-8]. Square brackets, [], are used to indicate a closed interval on the set \( \mathbb{R} \) of all real numbers. By definition, the closed interval \( [u, v] = \{x \in \mathbb{R} \mid u \leq x \leq v\} \) [58, pp. xv-xvii].

8 By symmetry, the maximum angles are the same for the items in the class plane labelled \( x_3 = -1 \).

9 By symmetry, the angle for this case will be the same as the for the case \( \mathbf{\tilde{c}'} = (0,1,1) \) and \( \mathbf{\tilde{d}'} = (0,1,-1) \).
In case ii), \( \cos \beta = \frac{(\mathbf{c}' \cdot \mathbf{d}')}{(|\mathbf{c}'| \cdot |\mathbf{d}'|)} = \frac{(1,1,1) \cdot (1,1,-1)/3}{1/3} = 30.5^\circ \).

So, for training cases from the different classes\(^{10} \), \( \beta > 70.5^\circ \). This gives \( \alpha \leq 60^\circ < 70.5^\circ < \beta \). Thus, in the expanded input space, the angle, \( \alpha \), between the training vectors from the same class will be less than the angle, \( \beta \), between the training vectors from different classes.

For vectors, \( \mathbf{x}, \mathbf{y} \), which have been normalized in length, we know that \( |\mathbf{x} \cdot \mathbf{y}| \) will be a constant value \( C \), so \( \mathbf{x} \cdot \mathbf{y} = C \cos \theta \). So for \( \mathbf{a}', \mathbf{b}', \mathbf{c}', \mathbf{d}' \) normalized : \( \mathbf{a}' \cdot \mathbf{b}' \geq C/2 > C/3 > \mathbf{c}' \cdot \mathbf{d}' \). For vectors normalized to unit length, \( C = 1 \), so \( \mathbf{x} \cdot \mathbf{y} = \cos \theta \). For \( \mathbf{a}', \mathbf{b}', \mathbf{c}', \mathbf{d}' \) normalized to unit length, \( \mathbf{a}' \cdot \mathbf{b}' \geq 1/2 > 1/3 > \mathbf{c}' \cdot \mathbf{d}' \). In either case, the inner product (i.e. the proximity) of vectors from the same class will always be greater than for vectors from different classes. So we see that, by either distance measure, any two vectors that share a class will always be closer to each other than to any vector from the other class.

Thus training vectors from different classes are mapped to two different class planes which are well-separated in the expanded input space. That is, not only do the two classes not overlap, but even when normalized, their separation is greater than the maximum spread of each class. This result generalizes directly to tasks where the elements of the formal input vectors have values in the interval \([0, R]\) where \( R > 1 \), as long as the values used to indicate the class label at the input are \( \pm R \). That is, the results also hold for \( l = 2 \), input labels \( a_{l+1}, b_{l+1} \in \{-c\} \) and where, \( \forall i \leq l, \exists a_i, b_i \in [0, R] \)\(^{11} \). This is the case since the bounding cases examined in the proof above would simply be scaled by \( R \). For example,

\[
\begin{align*}
\mathbf{a}' &= (1,1,1) \rightarrow (R, R, R) = R(1,1,1) = R\mathbf{a}' \\
\mathbf{b}' &= (1,0,-1) \rightarrow (R, 0, -R) = R(1,0,-1) = R\mathbf{b}'
\end{align*}
\]

And since scaling a vector changes only its magnitude, but not its direction, we have \( \forall \mathbf{x} \cdot \mathbf{y}, \)

\[
\cos \theta = \frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x}||\mathbf{y}|} = \frac{R\mathbf{x} \cdot R\mathbf{y}}{|R\mathbf{x}||R\mathbf{y}|}
\]

\(^{10}\)Since \( \mathbf{c} = \mathbf{d} \), this inequality is strict.

\(^{11}\)That is, \( \forall i, a_i \) and \( b_i \) are in the closed interval, \([0, R]\), which means that \( 0 \leq a_i \leq R \) and \( 0 \leq b_i \leq R \).
So the results computed above for the angles, $\alpha$ and $\beta$ as well as for the inner products of the normalized vectors also hold when $l=2$, with $a_{i+1}, b_{i+1} \in \{-R, R\}$ and where, $\forall i \leq l, a_i, b_i \in [0, R]^{12}$.

Further, even if the class labels were represented at the input by values of $\pm Q$, where $Q > R$, the classes would be well-separated, since increasing the magnitude of the class labels would only increase the distance between the class planes in the expanded input space. This decreases the angle between vectors in same class, while increasing the angle between vectors in different classes. The effect applies even when the vectors are normalized, since normalizing a vector changes its magnitude but not its direction.

For task b), because of the separation between classes discussed above, each node of the WTA(4) hidden layer (Figure 5.2) learns to respond to one of four sub-clusters (see Figure 5.5). In general, if there are $p \geq 2$ distinct subclusters in each class, then their centroids can be found by a hidden WTA layer of size $n = n^*p$, using, for example, Rumelhart’s and Zipser’s [67] method$^{13}$.

In WTA methods such as Rumelhart’s and Zipser’s [67], it is the separation between individual clusters (rather than between class planes) which is most essential. Showing that, in the expanded input space, the distance between class planes is greater than the maximum spread of either class planes (e.g. $\alpha < \beta$) ensures that the separation between individual clusters of training cases will be considerably greater than their maximum spread, as follows. By definition of a Cross-Dimensional Generalization task (Section 4.3.2), the subclusters of training data within each class plane of the expanded input space are distinct (i.e. well-separated)

$^{12}$Where $[0, R]$ denotes the closed interval.

$^{13}$In their method, Rumelhart and Zipser [67] show that a WTA cluster of $n$ nodes will converge to the centroids of $n$ well-separated clusters, each of which contains an approximately equal number of training cases. They use normalized input vectors, presented in random order with equal frequency. As in their use of the method for learning “unnatural” classifications, the number of subclusters must be the same for each class. To ensure that no competitive node is left out of the competition due to idiosyncrasies in the distribution of the data set or its order of presentation, they use ‘leaky learning’ [67]. Rumelhart and Zipser use, but do not specify, a mechanism whereby nodes ‘repel’ each other, to ensure that different nodes find different clusters.
from other clusters within that plane. That is, their separation will be somewhat greater than their maximum spread.

This means that the maximum cluster size will always be less than approximately 1/3 of the maximum spread of a class plane (e.g. $\alpha / 3$), as illustrated in Figure 5.6. Here, the shaded discs indicate the largest clusters that could be contained in a class plane and still have a separation that is at least as large as their maximum spread, $d$. In the context of methods that use inner product similarity measures, we can write this as $\delta < \alpha / 3$, where $\delta$ is the angle between the two most distant vectors in a cluster (since, trisecting the distance between two vectors trisects the angle between them).

Showing that, in the expanded input space, the distance between class planes is greater than the maximum spread of either class planes (e.g. $\alpha < \beta$), means that the separation between clusters from different class planes will be at least $\beta > \alpha > 3\delta$. So in the general case, even if the separation between class planes is only $\beta > \alpha/2 > 1.5 \delta$, all of the clusters in the expanded input space would be well-separated.

**Figure 5.6: Maximum Cluster Size Within a Class Plane**
So far, we have been considering tasks whose formal input space is two-dimensional (i.e. with \( l = 2 \)). For tasks with \( l > 2 \) elements in each formal input vector — depending on the relative values of \( l, R, \) and \( Q \) — the class planes may or may not be sufficiently separated in the expanded input space. The Appendix (Sections A.1.1 and A.1.2) describes the conditions under which the training data in the expanded input space would be well-separated according to class. However, tasks that do not initially satisfy this criterion could still be handled by the WTA method proposed here, by representing the input class labels by a larger value of \( Q \), according to equations derived in Sections A.1.1 and A.1.2. In particular, for networks that use inner product similarity measures, whether vectors are normalized or not, satisfying

\[
4Q^4 + Q^2lR^2 - l^2R^4 > 0
\]  

(5.4)

would be sufficient.

For networks that use Euclidean distance measures,

\[
Q \geq \frac{R \sqrt{l}}{2}
\]  

(5.5)

would be sufficient.

The remainder of this discussion includes binary classification tasks of the more general form (i.e. with \( l \geq 2 \) inputs whose values are in the closed interval \([0, R]\), and with input label values of \( \pm Q \). As long as \( Q \) is sufficiently large relative to \( l \) and \( R \) (according to 5.4 or 5.5, above) to ensure the necessary separation in the expanded input space, each node of the WTA(n) hidden layer (Figure 5.2) will learn to respond to one of the \( n \) sub-clusters.

The weights entering a node of the WTA(n) hidden layer correspond to the centroid of its cluster. In a Winner-Take-All cluster of nodes, only one node will win (i.e. have a value of 1) for any given input, while the other nodes in that cluster will have a value of zero. As long as the assumptions made by Rumelhart and Zipser [67] are satisfied, the winning node will be the one whose incoming weight vector is closest to the input vector. So by the end of training, the hidden layer will activate the node whose incoming weight vector is most similar to the input vector. That is, the node that represents the cluster whose centroid is nearest to the input
vector will be activated. For example, in task b), each of the four hidden WTA(4) nodes would correspond to one of the 4 subclusters of training cases. The incoming weight vectors of each node would be approximately equal to the coordinates of its cluster centroid. For example, in task b), the coordinates of the cluster centroids are (0.2, 0.8, 1), (0.25, 0.65, -1), (0.8, 0.35, -1), and (0.85, 0.15, 1) — see Figure 5.5.

The final two-node competitive layer can learn which centroid corresponds to which class label, since it receives the output of the WTA(n) hidden layer as well as the class labels, \(x_{l+1}\), directly from the input layer (see Figure 5.2). In particular, this is possible, because the inputs to the WTA(2) top layer also show clear clustering that is consistent with the classification, as follows:

In binary classification tasks (i.e. with \(m = 2\) classes), the vectors received by the WTA(2) output layer will have the form \((b_1, b_2, ..., b_n, x_{l+1})\), where \(b_1, b_2, ..., b_n\) are the outputs of the \(n\) nodes of the hidden WTA(n) layer nodes, and \(x_{l+1}\) is the input class label for that training case. Suppose, for notational simplicity, that the items in the class labelled, \(+Q\), (call this class A) activate the first \(n/m\) nodes of the WTA(n) layer. Let the ones with labels, \(-Q\), (calls this class B) activate the second set of \(n/m\) nodes of the WTA(n) layer. Then, we can denote the possible inputs to the WTA(2) layer as \((A_1, A_2, ..., A_{n/2}, B_1, B_2, ..., B_{n/2}, x_{l+1})\), since for binary classification problems, \(m = 2\), so \(n/m = n/2\).

Two training cases, \(a'\) and \(b'\), that belong to the same class will have the same class label (i.e. either \(a_{l+1} = b_{l+1} = +Q\) or \(a_{l+1} = b_{l+1} = -Q\). If they happen to be nearest to the same centroid within their class plane, then by the end of training they will both activate the same hidden layer WTA(n) node, and so provide identical vectors, \(\vec{e} = \vec{f}\), to the WTA(2) output layer. For example, if both training cases from the class whose input label is \(-Q\) are nearest to the centroid that activates the first node of the hidden WTA(n) layer, then \(\vec{e} = \vec{f} = (1, 0, 0, ..., 0, -Q)\). Since \(\vec{e} = \vec{f}\), the angle, \(\alpha\), between \(\vec{e}\) and \(\vec{f}\) is \(0^\circ\) (since they are identical) so \(\cos \alpha = \cos 0^\circ = 1\).

Otherwise, if the two training cases, \(a'\) and \(b'\) are in the same class but nearest to two different centroids, they will activate two different hidden nodes. For example, the inputs to the WTA(2)
output layer would be \( \overline{e} = (1, 0, 0, ..., 0, -Q) \neq \overline{f} = (0, 1, 0, ..., 0, -Q) \). So the inputs, \( \overline{e} \) and \( \overline{f} \), to the WTA(2) layer will differ by 1 in exactly 2 places. This gives

\[
\cos \alpha = \frac{\overline{e} \cdot \overline{f}}{|\overline{e}| |\overline{f}|} = \frac{(1, 0, ..., 0, -Q) \cdot (0, 1, ..., 0, -Q)}{|(1, 0, ..., 0, -Q)| |(0, 1, ..., 0, -Q)|} = \frac{0 + (-Q)^2}{\sqrt{1 + (-Q)^2} \sqrt{1 + (-Q)^2}} = \frac{Q^2}{(\sqrt{1 + Q^2})^2} = \frac{Q^2}{1 + Q^2}
\]

By inspection, this is always greater than 0. So in either case, for training inputs from the same class, \( \cos \alpha > 0 \) in the input space to the WTA(2) output layer.

Two training cases that belong to different classes will always be closest to different clusters in the expanded input space, since they are in separate class planes. So, by the end of training, they will activate different nodes in the WTA(n) hidden layer.

Thus, the outputs from the WTA(n) hidden layer will be orthogonal. Further, one case will have an input class label of \( x_{i+1} = +Q \), while the other case will have a class label of \( x_{i+1} = -Q \). For example, the inputs to the WTA(2) output layer would be of the form \( \overline{g} = (0, 0, ..., 1, 0, Q) \) and \( \overline{h} = (0, 0, ..., 0, 1, -Q) \).

So in the input space to the WTA(2) layer, vectors, \( \overline{g} \) and \( \overline{h} \), for items from different classes will be separated by an angle, \( \beta \), where
CHAPTER 5. TECHNIQUES

\[ \cos \beta = \frac{\vec{g} \cdot \vec{h}}{|\vec{g}| |\vec{h}|} \]

\[ = \frac{(0, 0, ..., 1, 0, Q) \cdot (0, 0, ..., 0, 1, -Q)}{|(0, 0, ..., 1, 0, Q)|| (0, 0, ..., 0, 1, -Q)|} \]

\[ = \frac{0 + (-Q)(Q)}{\sqrt{1 + (Q)^2} \sqrt{1 + (-Q)^2}} \]

\[ = \frac{-Q^2}{(\sqrt{1 + Q^2})^2} \]

\[ = \frac{-Q^2}{1 + Q^2} \]

By inspection, this is always < 0. And since we have shown that \( \cos \alpha > 0 \), we know that \( \cos \alpha > 0 > \cos \beta \).

Thus, for all \( Q, l \) and \( R \), \( \alpha < 90^\circ < \beta \). That is, in the input space of the WTA(2) layer, the angle, \( \alpha \), between training vectors from the same class will be less than the angle, \( \beta \), between training vectors from the different classes, for all \( l, R \), and \( Q > 0 \). Since normalization can change the magnitude, but not the direction of vectors, these results hold whether or not the vectors are normalized.

In practice, \( \alpha \) will be normally be significantly smaller than \( \beta \), since, by inspection of 5.6 and 5.7, we can see that only for very small \( Q \) does either \( \cos \alpha \) or \( \cos \beta \) approach 0. That is, only for very small \( Q \) will the values of \( \alpha \) and \( \beta \) approach 90\(^\circ\) (and hence, approach each other). For example, when \( Q = 1 \), \( \cos \alpha = 1/2 \) (or 1) and \( \cos \beta = -1/2 \), so \( \alpha = 60^\circ \) (or 0\(^\circ\)) and \( \beta = 120^\circ \). So \( \alpha \) is significantly smaller than \( \beta \). This discrepancy continues to grow as the value of \( Q \) increases since, by inspection of 5.6 and 5.7, we can see that, as \( Q \) increases, \( \cos \alpha \) increases toward +1, while \( \cos \beta \) decreases toward -1. Thus, \( \alpha \) decreases toward 0\(^\circ\), while \( \beta \) increases toward 180\(^\circ\).

\[ ^{14} \text{Note the change in the direction of the inequality. This is the case since, } \theta \text{ will be large when cos} \theta \text{ is small, and vice versa.} \]
toward 180°. For example, when \( Q = 2 \), \( \cos t = 4/5 \) (or 1) and \( \cos \beta = -4/5 \), so \( \alpha = 37° \) (or 0°) and \( \beta = 143° \).

So there is clear clustering of inputs to the WTA(2) layer that is consistent with the classification. Thus, the WTA(2) can find the centroids of these two clusters. In task b) of Table 5.1, these would be approximately \((0.5, 0.5, 0, 0, 1)\) and \((0, 0, 0.5, 0.5, -1)\), for example, if the two nodes of the WTA(4) layer that are activated by class 1 inputs happen to be the first two nodes of that layer.

Likewise, for networks that use Euclidean distance as a similarity measure, training inputs to the WTA(2) top layer will cluster according to class, since, the distance is either 0 (for identical vectors, \( \bar{e} \) and \( \bar{f} \)) or

\[
|\bar{e} - \bar{f}| = \sqrt{\sum_{i=1}^{l} (e_i - f_i)^2}
\]

\[
|\bar{e} - \bar{f}| = |(1, 0, ..., 0, -Q) - (0, 1, ..., 0, -Q)|
\]

(5.8)

\[
\sqrt{1^2 + (-1)^2 + (-Q + (-Q))^2} = \sqrt{2 + (-Q + Q)^2} = \sqrt{2}
\]

And for training items from different classes, the inputs to the WTA(2) layer, \( \bar{g} \) and \( \bar{h} \), will be separated by a distance of

(5.9)

\[
|\bar{g} - \bar{h}| = |(0, 0, ..., 1, 0, Q) - (0, 0, ..., 0, 1, -Q)|
\]

\[
= \sqrt{1 + 1 + (2Q)^2}
\]

\[
= \sqrt{2 + 4Q^2}
\]
So items from the same class will always be closer (either 0 or $\sqrt{2}$) than items from different classes, $(\sqrt{2}+4Q^2)$ in the input space to the WTA(2) layer. Only for very small values of $Q$ do $|\mathbf{e} - \mathbf{f}|$ approach $|\mathbf{g} - \mathbf{h}|$.

So far, the operation of the network has depended on similarity between entire training cases. In each item presented during training, the class labels are specified as part of the input vector. During testing, however, no class label is specified as part of the input to the network. Rather, novel test inputs are of the form $(x_1, x_2, ..., x_l, 0)$. That is, the class label is unspecified (i.e. neither $+Q$ nor $-Q$) when the novel test case is presented to the network. Recall that for test cases, an unknown class label is indicated by a value of $x_{l+1} = 0$, since this value provides no activation to the input label node. A value of zero is neutral (i.e. equally distant) relative to the values, $+Q$ and $-Q$, which are used to represent the input class labels in tasks of this form.

Generalization to novel test items will depend on the proximity of the novel inputs to the learned centroids of local clusters. Since the test case includes only zero values for the class labels, only the first $l$ elements of the test vectors (i.e. the formal inputs) can play a role in determining the nearest centroid. A zero value contributes nothing in inner product computations. And for models that use Euclidean distance, rather than the inner product as the similarity measure, a class label of 0 means that a test input is always equidistant from the two class planes.

This means the nearest centroid to the test case is determined by the first $l$ coordinates of the test case. Proximity of test cases to known cluster centroids thus must be determined in what is effectively the original input space. Nevertheless, generalization to novel test cases is still critically dependent on the similarity between training cases, since the relevant centroids could only have been learned by comparing the entire training cases, where the entire training case (including class label) is treated as a single input vector.

For example, the test case for task b) from Table 5.1 has coordinates $(0.4, 0.8, 0)$. In Figure 5.5 and Figure 5.7, the symbol ' ? ' indicates the projections of the test case onto the class planes. The cluster centroids of each subcluster are indicated by the symbol ' * '. The coordinates of the centroids for task b) are $(0.2, 0.8, 1)$, $(0.25, 0.65, -1)$, $(0.8, 0.35, -1)$, and $(0.85, 0.15, 1)$. This provides the grounds for generalization to an output of '1' for the novel test case,
since the test input is nearest to the centroid whose coordinates are \((0.2, 0.8, 1)\). Recall that Cross-Dimensional generalization includes only tasks that have the grounds for generalization described in Section 4.3.2. In particular, it must be stressed that any problem where the desired classification of the test case is not consistent with the label of the nearest class centroid would not be a case of Cross-Dimensional Generalization.

**Figure 5.7: Centroids of Local Clusters**

In the general case, when a test case is presented to the network, the WTA\((n)\) hidden layer computes the inner product of the input vector with the incoming weight vector for each of its nodes. The hidden node with the greatest inner product will win the competition\(^{15}\). Since the weight vector entering each of the WTA\((n)\) nodes stores the learned centroid of a cluster, the winning node corresponds to the centroid that is nearest to the input vector. The hidden WTA\((n)\) layer will activate only this single node, \(w\), with an activation value of 1, while the remaining nodes in the WTA\((n)\) layer will remain inactive.

\(^{15}\)For models that use Euclidean distance as a similarity measure, the winning node will be the one whose incoming weight vector is least distant from the test case.
In task b), for example (see Figure 5.5 and Figure 5.7), this would be the hidden node whose incoming weights are approximately \((0.2, 0.8, 1)\), which are the coordinates of the nearest cluster to that test case. This is the case, since the inner product of \((0.4 \ 0.8, 0)\) with \((0.2, 0.8, 1)\) = 0.72 will be greater than its inner product with any more distant centroid. This result would be the same whether or not the input vectors are normalized, since the angle between the vectors (and hence, their relative proximity as measured by the inner product) would be unchanged by normalization. In task b), the nearest centroid to the test case, as measured by Euclidean distance, is also \((0.2, 0.8, 1)\), since the Euclidean distance between \((0.4 \ 0.8, 0)\) and \((0.2, 0.8, 1)\) = 1.02 is less than its Euclidean distance from any other learned centroid.

In the general case, node \(w\) is the only one that is active in the input to the \(\text{WTA}(2)\) layer (since the label node for the test case is zero, and since the hidden layer will have exactly one winner). So the winning node in the \(\text{WTA}(2)\) layer will be the one that has the greatest incoming weight from node \(w\). Since node \(w\) corresponds to a cluster of uniformly-labelled training cases (i.e. from a single class plane from the expanded input space), only one of the \(\text{WTA}(2)\) nodes will have learned to respond to node \(w\). So the only \(\text{WTA}(2)\) node that will have significant weight on the connection from \(w\) will be the one that has learned to respond to the correct class label for that cluster. This is the node that will win the competition. Thus, the unlabelled novel test case is assigned the class label of its nearest centroid.

To see how this would occur, let us continue using the notational convention introduced above, where items in the class labelled, \(+Q\), activate the first \(n/2\) nodes of the \(\text{WTA}(n)\) layer and those with labels, \(-Q\), (calls this class B) activate the second set of \(n/2\) nodes. Then, we denote the possible inputs to the \(\text{WTA}(2)\) layer as \((A_1, A_2, \ldots, A_{n/2}, B_1, B_2, \ldots, B_{n/2}, x_{i+1})\). In task b), \(Q = 1\), and \(n = 4\) so the test input to the \(\text{WTA}(2)\) top layer is of the form \((A_1, A_2, B_1, B_2, 0)\). The winning hidden node, \(w\), for the test case \((0.4 \ 0.8, 0)\) has incoming weights \((0.2, 0.8, 1)\). This means that during training, node \(w\) learned to respond to training cases from the class plane labelled by \(+1\). Thus, the test input to the \(\text{WTA}(2)\) top layer is either \((1, 0, 0, 0)\) or \((0, 1, 0, 0)\). Recall that, during training, the \(\text{WTA}(2)\) output layer acquired weights of approximately \((0.5, 0.5, 0, 1)\) for the node that has learned to respond to training cases from Class A and
(0, 0, 0.5, 0.5, -1) for the node that has learned to respond to training cases from Class B. The inner product of these weights with the input to the WTA(2) layer give a result of 0.5 for Class A and 0 for Class B. Likewise, by Euclidean distance, the test input to WTA(2) is at a distance of approximately 1.22 from the weight vector (0.5, 0.5, 0, 0, 1), but its distance from (0, 0, 0.5, 0.5, -1) is approximately 1.58. So by either similarity measure, the WTA(2) output node that will be activated for the test case will be the one that learned to respond to test cases with from Class A (i.e. with class label = +Q = +1). Thus, the WTA(2) layer will produce the correct final output (i.e. by activating the WTA(2) output node that corresponds to the class labelled 1) in response to the test case (0.4 0.8, 0).

Returning to the general case, recall that the training inputs to the WTA(2) output layer were of the form \((0, 0, \ldots, 0, 1, 0, \ldots, \pm Q)\), where the only non-zero values were due to the single winning node, \(w\), of the hidden layer and the input class label. During training, each node of the hidden layer would have been activated (i.e. won) with approximately equal frequency, since the training set consists of \(p \geq 2\) subclusters per class, each containing approximately the same number of training cases. As shown above (5.6 to 5.9), the training inputs to the WTA(2) layer are well-separated into two uniformly-labelled clusters. Since in the notation we are using here, these are of the form \((A_1, A_2, \ldots, A_{n/2}, B_1, B_2, \ldots, B_{n/2}, x_{i+1})\), the training inputs to the WTA(2) node, \(u\), which has learned to respond to the class labelled +Q will be

\[
\begin{align*}
(1, 0, 0, \ldots, 0, 0, \ldots, 0, Q) \\
(0, 1, 0, \ldots, 0, 0, \ldots, 0, Q) \\
\vdots \\
(0, 0, \ldots, 0, 1, 0, \ldots, 0, Q).
\end{align*}
\]

where each of the first \(n/2\) nodes takes on a value of 1 approximately equally often. So the incoming weight vector of node \(u\) will be \(\overline{w}_u \approx (1/2, 1/2, \ldots, 1/2, 0, \ldots, 0, +Q)\), which is the centroid of its training inputs. Similarly, for the WTA(2) node, \(v\), which has learned to respond to training cases from the class labelled -Q, the weight vector will be \(\overline{w}_v \approx (0, \ldots, 0, 1/2, 1/2,\)
CHAPTER 5. TECHNIQUES

... 1/2, -Q). Finally, test inputs to the WTA(2) top layer will be of the form \( \mathbf{x} = (0, 0, ..., 0, 1, 0, ..., 0) \), where the only non-zero value is due to the single winning node, \( w \), of the hidden layer. If, during training, the uniformly-labelled subcluster associated with node \( w \) was in the class plane labelled \( Q \), then it will cause node \( w \) to win\(^{16}\), since \( \mathbf{x} \cdot \mathbf{w}_w \approx 1/2 \) and \( \mathbf{x} \cdot \mathbf{w}_v \approx 0 \).

For networks that use Euclidean distance as a similarity measure, the nearest centroid to the test case is \( \mathbf{w}_w \), since

\[
|\mathbf{x} - \mathbf{w}_w| = \sqrt{\frac{n}{2} + Q^2} < \sqrt{1 + \frac{n}{2} + Q^2} = |\mathbf{x} - \mathbf{w}_v|
\]

Otherwise, if the cluster of training cases corresponding to node \( w \) was in the class plane labelled \(-Q\), node \( v \) will win, since \( \mathbf{x} \cdot \mathbf{w}_w \approx 0 \) and \( \mathbf{x} \cdot \mathbf{w}_v \approx 1/2 \). For networks that use Euclidean distance as a similarity measure, the inequality in 5.10 is reversed, which means that node \( v \) will win. Thus, the test case will be assigned the same class label (i.e. activate the same output node) as the label of its nearest subcluster of training cases.

Classification into Multiple Distinct Classes

Although Clark and Thornton focus mainly on problems of classifying inputs into one of two classes, Cross-Dimensional Generalization techniques could be used to handle problems that classify inputs into more than two distinct classes\(^{17}\). The representational scheme exemplified in task c) of Table 5.1 could be used for this purpose. The (expanded) input vector presented to the input layer of a WTA network would be of the form \( (x_1, x_2, ..., x_l, x_{l+1}, x_{l+2}, ..., x_{l+m}) \), where \( m \geq 2 \).

---

\(^{16}\)Note that in Winner-Take-All networks, the similarity between an input vector \( \mathbf{x} \) and a WTA winning node's weight vector \( \mathbf{w} \) need not be high. It only needs to be greater than for any other WTA node in that layer. So even if \( p \) is very large, and so for the winning node, \( \mathbf{x} \cdot \mathbf{w} \approx 1/p \), this inner product would still be greater than those of every other node in that layer (which are approximately zero). In practice, the limits of the precision of the network would impose an upper limit on the size of \( p \) that could be handled effectively.

\(^{17}\)That is, since the classification to be learned separates the inputs into \( m \) distinct classes, no case would be labelled as belonging to more than one class.
In the most general case, we would have \( l \geq 2 \) inputs whose values are in the closed interval \([0, R]\), with input label values of \( Q \), and \( m \geq 2 \) classes. Then, \( x_{l+i} = Q \) when the training case is a member of class \( i \); otherwise, \( x_{l+i} = 0 \). For example, training inputs from the first class would include the class label, as in the vector \((x_1, x_2, \ldots, x_l, Q, 0, 0, \ldots, 0)\); inputs from the second class, as \((x_1, x_2, \ldots, x_l, 0, 1, 0, \ldots, 0)\) etc. Test inputs would be of the form \((x_1, x_2, \ldots, x_l, 0, 0, \ldots, 0)\), since no class label would be specified as part of the input vector.

The operation of the network is comparable to the binary classification tasks described above, except for some differences in the proofs for the separation of the training data. Section A.2 of the Appendix describes the conditions under which the training data in the expanded input space would be sufficiently separated. That is, for inner product similarity measures, we need

\[
Q^2 > \frac{lR^2}{\sqrt{2}}
\]

For Euclidean distance, we need

\[
Q > R\sqrt{\frac{l}{2}}
\]

As long as these are satisfied, the WTA(n) hidden layer will find the relevant subclusters, and the input to the WTA(m) top layer will be of the form \((b_1, b_2, \ldots, b_i, x_{l+1}, x_{l+2}, \ldots, x_{l+m})\), where \( b_1, b_2, \ldots, b_i \) are the binary outputs of the WTA(n) hidden layer, and the only \( i \)th class label has a non-zero value. For example, the input to the WTA(m) top layer would be \((b_1, b_2, \ldots, b_i, 0, Q, 0, \ldots, 0)\) for a training item from class \( i = 2 \).

Two training cases, \( \bar{a}' \) and \( \bar{b}' \) that belong to the same class will have the same class label. If they happen to be nearest to the same centroid within their class plane, then by the end of training they will both activate the same hidden layer WTA(n) node, and so provide identical vectors, \( \bar{e} = \bar{f} \), to the WTA(2) output layer. For example, the inputs to the WTA(2) output layer would be \( \bar{e} = (1, 0, 0, \ldots, 0, Q) \neq \bar{f} = (1, 0, 0, \ldots, 0, Q) \). Since \( \bar{e} = \bar{f} \), the angle, \( \alpha \), between \( \bar{e} \) and \( \bar{f} \) is 0° (since they are identical) so \( \cos \alpha = \cos 0° = 1 \).
Otherwise, if the two training cases, $\mathbf{a}'$ and $\mathbf{b}'$ are in the same class but nearest to two different centroids, they will activate two different hidden nodes. For example, the inputs to the WTA(2) output layer would be $\mathbf{e} = (1, 0, 0, \ldots, 0, \mathcal{Q})$ and $\mathbf{f} = (0, 1, 0, \ldots, 0, \mathcal{Q})$. So the inputs, $\mathbf{e}$ and $\mathbf{f}$, to the WTA(2) layer will differ by 1 in exactly 2 places. This gives the same result as for the binary case described above

$$
\cos \alpha = \frac{\mathbf{e} \cdot \mathbf{f}}{|\mathbf{e}| |\mathbf{f}|}
$$

$$
= \frac{(1, 0, \ldots, 0, \mathcal{Q}) \cdot (0, 1, \ldots, 0, \mathcal{Q})}{|(1, 0, \ldots, 0, \mathcal{Q})||(0, 1, \ldots, 0, \mathcal{Q})|}
$$

$$
= \frac{0 + \mathcal{Q}^2}{\sqrt{1 + \mathcal{Q}^2} \sqrt{1 + \mathcal{Q}^2}}
$$

$$
= \frac{\mathcal{Q}^2}{(\sqrt{1 + \mathcal{Q}^2})^2}
$$

$$
= \frac{\mathcal{Q}^2}{1 + \mathcal{Q}^2}
$$

By inspection, this is always greater than 0. So in either case, for training inputs from the same class, $\cos \alpha > 0$ in the input space to the WTA($m$) output layer.

Two training cases that belong to different classes will always be closest to different clusters in the expanded input space, since they are in separate class planes. Therefore, by the end of training, they will activate different nodes in the WTA($n$) hidden layer. Thus, the outputs from the WTA($n$) hidden layer will be orthogonal. Further, one case will have an input class label of $x_{i+j} = \mathcal{Q}$, while the other case will have a class label of $x_{i+k} = \mathcal{Q}$. (Note the indices here: $j \neq k$, since the training cases are in different classes). This means that the inputs to the WTA($m$) top layer will also be orthogonal. For example, the inputs to the WTA(2) output layer would be of the form $\mathbf{g} = (0, 0, 1, 0, 0, \ldots, 0, \mathcal{Q})$ and $\mathbf{h} = (0, 0, 0, 1, 0, \ldots, \mathcal{Q}, 0)$. 
So in the input space to the WTA($m$) layer, vectors $\mathbf{g}$ and $\mathbf{h}$, for items from different classes will be separated by an angle, $\beta = 90^\circ$, and $\cos \beta = 0$. And since we have shown that $\cos \alpha > 0$, we know that $\cos \alpha > 0 = \cos \beta$.

Thus, for all $Q$, $l$ and $R$, and $m$, $\alpha < 90^\circ < \beta^{18}$. That is, in the input space of the WTA($m$) layer, the angle, $\alpha$, between training vectors from the same class will be less than the angle, $\beta$, between training vectors from the different classes, for all $l$, $R$, and $Q > 0$. Since normalization can change the magnitude, but not the direction of vectors, these results hold whether or not the vectors are normalized.

In practice, $\alpha$ will be normally be significantly smaller than $\beta$, since, by inspection of 5.13, we can see that only for very small $Q$ does $\cos \alpha$ approach 0. That is, only for very small $Q$ will the values of $\alpha$ approach the value of $\beta = 90^\circ$. For example, when $Q = 1$, $\cos \alpha = 1/2$ (or 1) so $\alpha = 60^\circ$ and $\beta = 90^\circ$. So $\alpha$ is significantly smaller than $\beta$. This discrepancy continues to grow as the value of $Q$ increases since, by inspection of 5.13, we can see that, as $Q$ increases, $\cos \alpha$ increases toward $+1$, while $\cos \beta$ remains at 0. Thus, $\alpha$ decreases toward $0^\circ$, while $\beta$ remains at $90^\circ$. For example, when $Q = 2$, $\cos \alpha = 4/5$ (or 1), so $\alpha = 37^\circ$ (or $0^\circ$) and $90^\circ$.

Likewise, for networks that use Euclidean distance as a similarity measure, training inputs to the WTA($m$) top layer will cluster according to class, since, the distance is either 0 (for identical vectors, $\mathbf{e}$ and $\mathbf{f}$) or

$$|\mathbf{e} - \mathbf{f}| = \sqrt{\sum_{i=1}^{l} (e_i - f_i)^2}$$

$$|\mathbf{e} - \mathbf{f}| = |(1, 0, \ldots, 0, Q) - (0, 1, \ldots, 0, Q)|$$

(5.14)

$$\sqrt{1^2 + (-1)^2 + (Q - Q)^2} = \sqrt{2 + 0} = \sqrt{2}$$

---

18 Note the change in the direction of the inequality. This is the case since, $\theta$ will be large when $\cos \theta$ is small, and vice versa.
And for training items from different classes, the inputs to the WTA\((m)\) layer, \(\mathbf{g}\) and \(\mathbf{h}\), will be separated by a distance of

\[
|\mathbf{g} - \mathbf{h}| = |(0, 0, 1, 0, 0, \ldots, 0, Q) - (0, 0, 0, 1, 0, \ldots, Q, 0)|
\]

\[
= \sqrt{1^2 + (-1)^2 + (Q)^2 + (-Q)^2}
\]

\[
= \sqrt{2Q^2}
\]

\[
= Q\sqrt{2}
\]

So to ensure adequate separation of training cases at the WTA\((m)\) output layer, we would want to choose \(Q\) somewhat larger than 1.

During testing, the operation of the network would be identical to the case for the binary classification tasks. Since test cases contain only values of zero as input class labels, similarity is determined entirely by the first \(l\) vector elements (i.e. the formal inputs). Note that in Winner-Take-All networks, the similarity between an input vector \(\mathbf{x}\) and a WTA winning node's weight vector \(\mathbf{w}\) need not be high. It only needs to be greater than for any other WTA node in that layer. So even if \(m >> l\) (and so the vast majority of the elements of the test input vector \(\mathbf{x}\) would be zero)\(^{19}\), the inner products of the test input with each of the weight vectors of the nodes any given layer of would still be greatest for the node whose centroid is nearest, than it would be for any other nodes in that layer.

**Approximation of Training Data**

As is the case with Standard First-Order generalization using back-propagation, generalization performance by Winner-Take-All networks depends on how well the pattern of clustering of

\[^{19}\text{That is, } m >> l \text{ when the number of (nodes used to represent the) class labels is much greater than the dimensionality of the original input space.}\]
the training data is captured by the network. Architectural details, such as the size of hidden layers, affect how well the network can approximate the training data\textsuperscript{20}. For example, in the sample tasks from Table 5.1 which have been used in this thesis to illustrate the WTA method, each class was divided into only a small number of subclusters (i.e. $p = 2$). In more complex cases, such as Thornton’s TFT example (see Figure 5.8), a larger hidden layer would be needed in order to find the large number of subclusters in within each class.

\textbf{Figure 5.8: Clustering of Thornton’s Training Cases}

\hspace{1cm}

\textsuperscript{20}Rumelhart’s and Zipser’s [67] method is particularly sensitive to how well the number of nodes in the competitive layers corresponds to the number of uniformly-labelled subclusters in the training set. A potentially more robust competitive learning implementation might employ Kohonen’s Self-Organizing Maps (SOM’s) [45], since, rather than a one-to-one mapping between classes and WTA nodes, similar patterns in the input space are mapped to a ‘topological neighbourhood’ of several nodes which blend into other neighbourhoods [42, pp. 448-449].
CHAPTER 5. TECHNIQUES

One way to evaluate how well the network has approximated the training data is to test the network on the known, but now unlabelled, training cases. The more consistent the labels assigned by the network during this type of testing are with the actual input labels used during training, the better the approximation. For binary classification tasks that could be approached by the WTA technique for Cross-Dimensional generalization, we can expect fewer than approximately 11% of unlabelled training cases to be misclassified. Details of the error estimation are provided in Section A.3 of the Appendix. This performance is competitive with Thornton's TFT strategy [76]. Although he does not report the numerical data for his training inputs or his test cases, Thornton reports a 73% generalization accuracy in his TFT trials [74].

Finally, the performance of the Winner-Take-All technique can be compared with that of Back-Propagation networks. Although the performance of both Back-Propagation and Winner-Take-All networks depends on how well the details of the network architecture support the approximation of the training data, Winner-Take-All networks would appear to exploit the various levels of clustering in ways that Feedforward Back-propagation networks may not. Section 5.1.2 section explores this possibility and its relationship to the property of training independence.

5.1.2 Training Independence

Training independence has been identified by Marcus [52, 53] as a key factor in the inability of feedforward back-propagation networks to perform Universal Generalization. Recall from Chapter 2 that, by Marcus' definition, a network exhibits output independence when what one output node learns is independent of what any other output node learns. Marcus argues that this (as well as input independence) is the case for feedforward back-propagation networks, regardless of the number of nodes or layers (see Section 2.1.4).

But in the case of WTA networks, what each output node learns does depend on what the other output nodes learn. Equation 5.1 on page 92 shows that the output $b_i$ of node $i$ in the competitive layer depends, not only on its own connections to the input layer, but also on the inputs received by every other output node. These depend on the weights entering each
competitive node, hence, on what has been learned by those nodes. This interdependence is due to the lateral inhibition between output nodes in the competitive layer (see Figure 5.1). So Winner-Take-All networks do not exhibit output independence.

Further, in competitive learning, a neuron learns by shifting synaptic weights from its inactive to its active input nodes. This means that the change in the weight coming out of one input node does depend on the activation of other input nodes. Thus, Winner-Take-All networks also do not exhibit input independence.

A key factor in the capacity of WTA networks to perform cross-dimensional generalization is that there is useful learning across nodes. Nodes in the competitive layers communicate with each other to represent separate subclusters of uniformly-labelled cases in the (expanded) input space. During training, each input vector is treated as a whole, in that the weight vector of the winning node moves toward that point in its input space. The importance of this interaction in WTA networks suggests that networks exhibiting training independence, such as feedforward back-propagation networks might be less successful than WTA as techniques for Cross-dimensional generalization.

One of Thornton's main results [73, 76] is that feedforward back-propagation networks perform poorly on problems with low geometric separability of inputs. This would be the case for the task shown in Table 5.1 a), since it has a relatively low GSI. So even though the output of each training case is actually available to the network as the target value used in the weight update rule, back-propagation networks would not be able to exploit the pattern of vector similarity between training cases that can support Cross-Dimensional Generalization.

However, the task that is presented to the WTA network (shown in Table 5.1 c) includes the class labels as part of the training vectors presented to the input layer. In this expanded input space, the GSI of the problem is high. This would suggest that back-propagation networks might have more success performing the task in Table 5.1 c) than the one originally specified in Table 5.1 a).

But even if class labels were also included as part of the training vectors presented at the input layer (as well as being available as target values during weight training) of a feedforward
CHAPTER 5. TECHNIQUES

back-propagation network, training independence suggests that there might be insufficient learning across nodes to support Cross-Dimensional generalization. In particular, since the class labels (i.e. $x^3$ and $x^4$ from Table 5.1 c), which are presented at the input layer during training, are perfectly predictive of the outputs, the network could learn the training data perfectly without learning anything about the first two elements of the input (i.e. $x^1$ and $x^2$ from Table 5.1 c). That is, learning need only occur on the weights from these nodes.

My simulations on the tasks in Table 5.1 b) and c) suggest that this is the case\(^{21}\). Feedforward networks trained by back-propagation were able to learn the training cases, but they could not generalize to any novel test cases that did not include the class labels as input — such as the test cases from Table 5.1.

Further, they could not generalize to any test cases that were identical to the known training cases, but in which the class label was unspecified. For example, if after training on the labelled training cases in Table 5.1 c), the inputs of any of those same training cases are presented to the network without the class labels — e.g. (0.8, 0.1, 0, 0) — the network responds with output values between approximately 0.4 and 0.6. Even if we accept a very large margin of error in the magnitudes of the outputs — e.g. where the output (0.604, 0.389) would be interpreted as (1, 0), and (0.496, 0.511) as (0, 1) — the output is consistent with the correct class label only 50% of the time. That is, the performance is no better than chance. Varying the number and size of hidden layers did not alter this effect.

By contrast, in my simulations of the Winner-Take-All network shown in Figure 5.1 on the tasks in Table 5.1 b) and c), the network was able to generalize correctly to both the known training cases (i.e. when they were presented as test cases without labels), as well as to novel test cases (which also were presented without labels)\(^{22}\) that fall within the scope of cross-dimensional generalization tasks. For example, after training on the data shown in Table 5.1,

\(^{21}\)Trials used the standard back-propagation algorithm implemented in the Stuttgart Neural Network Simulator (SNNS) Version 4.2 [82]. Feedforward architectures with 0, 1, and 2 hidden layers were tested. For each of these, various, but not exhaustive, combinations of hidden layer sizes (up to 8 nodes per layer) were tested. Trials used learning rates of 0.01 to 2 with randomly initialized weights and input patterns presented in random order for up to 100,000 epochs.
when presented with the novel, unlabelled test inputs (0.4, 0.8), (0.7, 0.7), (0.9, 0.4), (0.6, 0.1), (0.2, 0.5), (0.1, 0.9), the network produced the class label of the centroid that was nearest to each test of these test cases in the original input space (i.e. 1, 0, 0, 1, 0, and 1, respectively).

These simulations suggest that Winner-Take-All Networks have an advantage over feedforward back-propagating networks in performing Cross-Dimensional Generalization tasks. When the training labels are included at the input layer of the back-propagation networks, they can learn the training cases. But they are unable to generalize to novel or even ‘known’ items from the training set, where the class label is not included as part of the test input. Thus, even when, during training, the labels are presented at the input layer, feedforward networks trained by back-propagation are less able than Winner-Take-All networks to exploit the clustering present in the expanded input space.

5.2 Relational Trace Techniques

The class of Core Relational Generalization (Section 4.4.1) includes tasks that have played a central role in arguments about the flexibility of human generalization abilities. In particular, Phillips’ Cross-Category and Non-Category generalization tasks (Section 2.3.2) and Marcus’ Binary Identity (Section 2.1) and Sentence Completion tasks (Section 2.1.2) have been presented as key examples of how humans can generalize to the most radically novel test cases. These researchers have observed that eliminativist methods such as back-propagating feedforward networks cannot solve these tasks, largely due to the problem of training independence.

---

22The network was implemented in MATLAB based on the Competitive Learning algorithm of the Neural Network Toolbox [17]. The network uses a standard conscience mechanism [18, 17] to ensure that no competitive node is left out of the competition due to idiosyncrasies in the distribution of the data set or its order of presentation. In Rumelhart’s and Zipser’s method, this was accomplished using ‘leaky learning’ [67]. Trials used a learning rate of 0.01 with a conscience learning rate of 0.001. Weights were randomly initialized and input patterns were presented in random order for 500 epochs.
But, as suggested in Section 4.2.2, there is reason to believe that learning (i.e. weight change) may not be the immediate mechanism by which humans perform these tasks. This section shows how Core Relational Generalization could be achieved using node activation, rather than weight change, to extract the relational information from the 'training' cases in such tasks.

The method involves computing the relational trace of a task. A relational trace is an array which encodes similarities between elements of the training cases. Since this information is abstracted from the individual feature values of the training cases, generalization to radically novel training cases (i.e. those whose outputs fall outside the training space) is possible.

An architecture for a Relational Trace Network, based on Winner-Take-All networks, is introduced in Section 5.2.1. Section 5.2.2 describes how the network could support Core Relational Generalization. The relational trace method and architecture are among the original contributions of this thesis.

5.2.1 Architecture

The relational trace network is composed of a number of trace modules, each of which consists of a bank of relational units. Each relational unit uses a Winner-Take-All network to compute the exclusive-or (XOR) function of its two binary inputs: \((x_i, y_i)\). The output \(u_i\) of a relational unit (Figure 5.9) is simply the exclusive or (XOR) function of its binary scalar inputs \((x_i, y_i)\). By computing the XOR of its inputs, the unit determines whether they are the same or different. The output \(u_i\) will be near 1 when \(x_i\) and \(y_i\) are different; when they are the same, \(u_i\) will be near 0.

Although the XOR operator can also be implemented by two layers of standard perceptrons (e.g. [59, 42, p. 177, 53, p. 16]), Winner-Take-All networks are considered to be more biologically plausible than Back-propagation networks [53, p. 29] and have been shown to be significantly more computationally powerful than perceptrons [50, 51]. Further, the XOR function has been shown to be learnable in biologically plausible ways by Winner-Take-All networks [46, 11]. For the most strictly biologically learnable versions, see Klemm et. al. [46].
Figure 5.9: Relational Unit

\[ u_i = \text{XOR}(x_i, y_i) \]

inputs: \( x_i, y_i \)

Figure 5.10 shows how relational units are assembled within a trace module. The module takes as input two binary vectors \( \bar{x} = (x_1, x_2, \ldots, x_n) \) and \( \bar{y} = (y_1, y_2, \ldots, y_n) \). Inside a trace module, the relational units are arranged in a linear array which computes the element-wise XOR of the input vectors. The outputs of the individual units are summed\(^{23}\) at \( q \), such that:

\[
q = \sum_{i=1}^{n} u_i
\]  

(5.16)

The result is that \( q \approx 0 \) when the input vectors \( \bar{x} = (x_1, x_2, \ldots, x_n) \) and \( \bar{y} = (y_1, y_2, \ldots, y_n) \) are identical. But if even one relational unit detects a difference between its inputs (e.g. \( x_i \neq y_j \)) then \( u_j \approx 1 \), so \( q \approx 1 \) (since nodes in this model take on values in the range of \([0..1]\)). Thus, the activation value of \( q \) will be near 0 whenever the input vectors \( \bar{x} \) and \( \bar{y} \) are identical; otherwise, it will be near 1.

The values computed within a trace module are intimately linked to the grounds for generalization in core relational tasks. Recall that Core Relational Generalization depends on observing a pattern of similarity (identity) between the output and elements of the input within each training case (see Section 4.4.1). The more consistently this is observed over the training set, the stronger the grounds for generalization.

\(^{23}\) Unlabelled connections shown in diagrams have a weight of 1. Throughout this model, nodes take on activation values in the closed interval \([0, 1]\).
This pattern of similarity between inputs and outputs is what an assembly (see also the discussion, later in this section, of Figure 5.1) of trace modules computes. A single relational trace module computes one element of the relational trace array described in Section 4.4.1. The value $q$ is the value of the relational trace element for the current training case.

**Figure 5.10: Architecture of a Trace Module**

**Output:**

$r(t) = \text{ scalar relational trace value at time } t$

**Input:**

Binary Vectors $\bar{x}, \bar{y}$

$\bar{x} = (x_1, x_2, x_3, \ldots, x_n)$

$\bar{y} = (y_1, y_2, y_3, \ldots, y_n)$
In order to determine how consistently similar or different its inputs have been over time, each trace module includes a feedback loop to form a memory of previous outputs of the WTA relational units. The purpose of the memory is to incorporate the results of previous 'training' cases into the current output, \( r \), of the trace module.

The output \( r(t) \) of the filter, at time \( t \), is described by the following equation\(^24\) [42, pp. 18-20]:

\[
    r(t) = \sum_{k=0}^{\infty} w^{k+1} q_i(t-k)
\]

The feedback system is stable for any fixed weight, \( w \), in the range \( 0 < w < 1 \) [42, p. 20]. The higher the value of \( w \), the slower the exponential decay of the memory\(^25\). With a weight of \( w = 1 \), there is no decay (i.e. the memory is perfect)[42, p. 20].

The value of \( r \) reflects how consistently similar or different \( \bar{x} \) and \( \bar{y} \) have been over recent cases. High values of \( r \) indicate that \( \bar{x} \) and \( \bar{y} \) have been different in at least some recent cases. Low values of \( r \) indicate that \( \bar{x} \) and \( \bar{y} \) have been consistently identical in recent cases.

In its most general form (see Section 4.4.1), a relational trace is a two-dimensional array. As shown in Figure 5.11 and Table 5.2, only a portion (i.e. the shaded region) is needed for Core Relational Generalization.

**Figure 5.11: Portion of Relational Trace for Core Relational Generalization**

\(^{24}\)The filter in Figure 5.10 consists of the nodes labelled \( q, q' \), and \( r \) and the connections between them. This architecture corresponds to the standard filter described in [42, pp. 18-20]. The rest of the architecture of the trace module introduced in Figure 5.10 is original.

\(^{25}\)Weights greater than about 0.9 would produce useful results for problems with few training cases.
**Table 5.2: Similarity Matrix**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>rose</th>
<th>is</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>rose</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>is</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 5.12: Architecture for Integrating Multiple Trace Modules**

\[ \bar{y} = (y_1, y_2, \ldots, y_n) \]

\[ x = (x_1, x_2, \ldots, x_n) \]
Figure 5.12 shows how a linear array of trace modules could be combined to form a network that supports Core Relational Generalization. Training cases are presented to the network one at a time. The $j^{th}$ module compares the $j^{th}$ word of the training case (e.g. for $j = 3$, $\overline{x}$ would be the vector representation of it) with the output of that case (e.g. for each module, $\overline{y}$ is rose).

The scalar value, $r_j$, which is output from module $j$, carries information about how the module's inputs, $\overline{x}$ and $\overline{y}$, are related to each other. Recall that high values of $r$ indicate that $\overline{x}$ and $\overline{y}$ have been different in at least some recent cases. Low values of $r$ indicate that $\overline{x}$ and $\overline{y}$ have been quite consistently identical in recent cases. In effect, the output, $r_j$, of module $j$ flags whether or not the input $\overline{x}$ to that module has been consistently identical to (hence, predictive of) the output $\overline{y}$ in recent cases. Recent differences are flagged by values of $r_j$ near one; consistent identity, by values near zero.

This information can be used to determine which inputs are useful in generating a prediction for the output of the new test case. The architecture shown in Figure 5.12 makes use of the $r_j$ values in generating (for each module) a prediction vector, $\overline{p} = (p_1, \ldots, p_n)$, where $p_i = x_i - r_j$. Figure 5.13 shows a more detailed view of the connectivity of the relevant nodes.

**Figure 5.13: Connectivity of Prediction Vector**

---

26 As discussed in Section 4.4.2, the information in a relational trace may also be relevant to other types of tasks. To the extent that such tasks depend on the information in a relational trace array, a two-dimensional array of trace modules might be helpful as a key component of connectionist solutions.

27 Recall from the definition of Core Relational generalization that a defining characteristic of such tasks is 'similarity (identity) between the output and components of the input within each case' (Section 4.4.1). Tasks in which the 'output' is not the final word of the sentence might be considered to be linking problems. Ways in which relational trace methods might play a role in their solution are discussed in Section 4.4.2.
The negative weights applied to the output of each module have the effect of subtracting the ‘flag’ from each element of the original binary input vector (e.g. word), $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. That is, $p_i = \mathbf{x}_i - r_j$. Consider the case where $r_j \approx 1$, since the inputs to module $j$ have been flagged as not useful in generating a prediction. Subtracting a value of $r_j \approx 1$ from each $x_i$ (where $0 \leq x_i \leq 1$) gives a prediction vector of $\mathbf{p} = (0, 0, \ldots, 0)$. So modules that report dissimilarity between their inputs will have prediction vectors whose elements are all near zero.

In the alternative case, where $\mathbf{x}$ and $\mathbf{y}$ have been consistently identical in recent cases presented to that module, $r_j$ will be near zero. Subtracting $r_j \approx 0$ from each element of $\mathbf{x}$ gives $\mathbf{p} \approx (x_1, x_2, \ldots, x_n)$. So a module whose inputs have been consistently identical in recent cases will have a prediction vector, $\mathbf{p} \approx \mathbf{x}$.

The final output of the network is the element-wise sum $\mathbf{z} = (z_1, \ldots, z_n)$ of the prediction vectors. Since the prediction vectors from modules whose inputs have been dissimilar are near zero, they contribute little to the sum. But modules whose inputs were consistently identical to the training output will have a prediction vector, $\mathbf{p} \approx \mathbf{x}$. So the final output $\mathbf{z}$, of all the prediction vectors will also be approximately $\mathbf{x}$.

Section 5.2.2 illustrates the operation of the network for sample tasks in the class of Core Relational Generalization.

### 5.2.2 Applications

The relational trace network could address Core Relational Generalization tasks of the form of Marcus’ Binary Identity and Sentence Completion tasks (Figure 5.12) as well as Phillips’ Cross-Category and Non-Category Generalization examples.

For the Sentence Completion task, the representational schemes Marcus seems to have in mind require the representation of a word, phoneme, or distinctive feature by a single node\(^\text{28}\). The size of the modules depends on which of these alternative representational schemes is the case. For example, in a scheme using one distinctive feature per node, since a phoneme is

---

\(^{28}\) Marcus acknowledges [54] that one word per node is unlikely to be biologically realistic representational scheme.
composed of about 15 independent distinctive features [3, p. 340], each phoneme could be represented by a binary vector of 15 nodes. That is, each of the 15 nodes would correspond to a particular distinctive feature. The value of that node would indicate the presence (a value of 1) or absence (a value of zero) of that feature in the phoneme. If a word is a sequence of up to \( m \) phonemes, where each phoneme is represented by a binary vector of size 15 (one node per distinctive feature), then each word could be represented by a binary vector of size \( n = m \times 15 \). Each module would then consist of \( n \) units.

Alternatively, if the data were presented visually, a word could be represented as a sequence of up to \( n \) graphemes, where each is represented by 26 nodes (i.e. one for each letter of the alphabet). The identity of the grapheme would be determined by which one of the 26 nodes is activated. For example, the grapheme ‘b’ would be represented by the vector \((0, 1, 0, 0, ..., 0, 0)\). Then, if a word is a sequence of up to \( m \) graphemes, it could be represented by \( n = m \times 26 \) nodes and each module would consist of \( n \) units.

Both of these ground form interpretations preserve the key properties of universal generalization and core relational generalization. In each of these cases, the task form is one of generation, rather than classification of the UQOTOM between inputs and outputs. Novel test outputs lie outside of the training space of the task. That is, they lie outside of the training space of the ‘training’ cases Marcus presents for the task: \( a \) rose is a rose, \( a \) tulip is a tulip, \( a \) lily is a lily. For instance, in the graphemic representation scheme, \textit{blicket} activates the nodes for \( b \), \( c \), and \( k \), whereas the ‘training’ cases do not. In the phonetic scheme, \( b \) activates the node for the distinctive feature value +labial, whereas the letters previously encountered in that position (\( r \), \( t \), and \( l \)) have the feature value -labial [1, pp. 94-95].

Both of these representational schemes are among those which are workable for a relational trace network. In either case, the number of modules required is equal to the number of words in a training or test case. Even for relatively long training cases, such as \textit{John painted the chair aka. What colour is the chair?} from Phillips’ non-category example, a bank of ten modules

\[29\] In a word with \( k \) phonemes, where \( k < m \), only the first \( 15 \times k \) phoneme positions would be activated. The remaining \( m-k \) elements of the word vector have a value of zero. A comparable situation would be the case for the graphemic representations discussed in this section.
(one per word) would be sufficiently large. In Marcus’ task, only the first four of these would be active.

Figure 5.12 shows how a training case (i.e. *a rose is a rose*) of Marcus’ Sentence Completion task would be presented to the network. Each trace module compares the final output with a word from the input. Consider the case where a module compares two identical words (e.g. the second and final words of the case *a rose is a rose*). When this case is presented to the network, as in Figure 5.12, the second module performs a comparison of the second word (i.e *rose*) with the final word, which is also *rose*. Table 5.3 shows the effect of the second module, whose inputs, *rose* and *rose*, are (entirely) similar. The diagonal shows the outputs ($u_1, u_2, \ldots, u_n$) of the relational units within that module.

Here, the graphemic representational scheme is used. For purposes of illustration, assume that the alphabet consist of the letters \{b, e, i, l, o, r, s\}. Then the letter *r* would be represented by the vector \(0, 0, 0, 0, 0, 1, 0\). The diagonal elements in the table show the results of the internal XOR comparisons made by the units within the module. When the binary inputs $x_i$ and $y_i$ are the same, the value at $u_i$ (see Figure 5.9) will be ‘0’. If they are different, then $u_i = 1$. Since the inputs, *rose* and *rose*, are identical, the output of each unit will be zero. Within the module (see Figure 5.10), these are summed, to give a value of zero at $q$. In the training cases that follow, this module compares *lily* with *lily*, and *tulip* with *tulip*. In each case, the sum of the units outputs will be zero.

When inputs to a module are dissimilar, however, node activations that are not common to both inputs will be flagged by the relational units as $u_i = 1$. Table 5.4 shows how the third module of Figure 5.12 compares the input word *is* with the output word *rose*. The diagonal of the table shows where the relational units whose input activations differ will produce an output of 1. Within the module (see Figure 5.10), these are summed, to give a value of $q = 1$.

One way to understand the computation performed by the modules is to consider how the $q_i$ values computed within each of the modules relate to the similarity matrix of a training case. The vector composed of these values, $\vec{q} = (q_1, q_2, \ldots, q_I)$, where $I$ is the number of words in a
training case) can be seen to equal the crucial portion of the similarity matrix for the training cases (i.e. the shaded regions of Table 4.5 and Table 4.6).

In order to take into account how consistently this pattern is present in the training set, each module incorporates an exponentially decaying memory of previous training cases. The output $r_i$ of trace module $i$ is generated according to the equation [42, pp. 18-20]:

$$r_i(t) = \sum_{k=0}^{\infty} w^{k+1} q_i(t-k)$$

(5.18)

The vector composed of these values, $\mathbf{r} = (r_1, r_2, \ldots, r_l)$, is the relational trace. That is, it is an array whose elements indicate the pattern of internal similarity that is relatively stable over the training set. For example, a weight$^{30}$ of $w = 0.95$ corresponds to a memory where useful information is retained for approximately 7 training cases. From 5.18 above, for $w = 0.95$,

$$r_i(t) = 0.95q_i(t) + 0.9q_i(t-1) + 0.86q_i(t-2) + 0.81q_i(t-3) + 0.77q_i(t-4) + 0.74q_i(t-5) + 0.7q_i(t-6)$$

(5.19)

In response to the first training case, the $i^{\text{th}}$ module, will thus produce an output of $r_i = 0.95$ if its inputs (i.e. the two words it compares) are different (since $q_i = 1$). If its inputs are the same, $q_i = 0$, so $r_i = 0$.

When each subsequent training case is presented, $r_i$ incorporates the new value of $q_i$ by adding it to the decaying memory of its previous values. In the Sentence Completion task, the first, third, and fourth modules consistently (i.e. for every training case) encounter differences between their inputs (i.e. the words they compare). This means that for each of these modules, $q_i = 1$, so $r_i = 0.95$ after the first training case. After the second, $q_i = 1$, so $r_i = 0.95 + 0.9$. Since nodes have a maximum activation of 1, the output of the module will be $r_i = 1$. For the third case, $q_i = 1$, so $r_i = 0.95 + 0.9 + 0.86$, so the module again outputs a value of $r_i = 1$. This process continues in this manner for subsequent cases presented to these modules.

On the other hand, the second module in the Sentence Completion task never encounters a difference between its inputs. This means $q_i$ is always zero, so the module outputs $r_i = 0$.

$^{30}$Any fixed weight near 1 would work well. A weight of $w = 1$ would also work. This would produce more accurate — but perhaps less realistic — results, since the memory would have no decay.
However, if a module encounters a difference between its input for even one training case, the difference will be flagged by a high value of $r_i$. And the value of $r_i$ will continue to be relatively high for subsequent cases (subject to decay, for $w < 1$). For example, suppose that for the first few training cases, the inputs to the $i^{th}$ module encounters (i.e. the words it compares) are the same. Then for each case, the module's values of $q_i$, and hence $r_i$ will be consistently near zero. But, crucially, if the module next encounters a training case in which the words it compares are different, it will generate a value of $q_i = 1$ and so $r_i = 0.95$. The value of $r_i$ will continue to be relatively high for subsequent training cases, even if no further discrepancies between the input to the module are encountered. That is, for the first case after the difference was flagged, $q_i = 0$, so $r_i = 0.95(0) + 0.9(1) = 0.9$. For the next case, $q_i = 0$, so $r_i = 0.95(0) + 0.9(0) + 0.86(1) = 0.86$.

Thus, differences that appear in even one training case will be reflected by a relatively high value of $r_i$ for several subsequent cases. The persistence of this effect will depend on the decay rate of the memory.

Finally, when the test case is presented, the value of the most recently calculated $r_i$ is used to determine a prediction for each module. For example, in the test case a blicket is a ..., the first, third, and fourth modules will have high values for $r_1$, $r_3$, and $r_4$, respectively. Recall that the prediction of the $i^{th}$ module is $\overline{p} = (p_1, p_2, ..., p_5)$, where $p_j = x_j \cdot r_i$. A high value of $r_i$ means that the prediction, $\overline{p}_i$, will be a vector of values near zero. However, the for the second module, $r_2$ will be near zero, so $p_j = x_j \cdot r_2 \approx x_j \cdot 0$. So the prediction vector is $\overline{p} = \overline{x}$, which, for the second module is the vector that represents blicket.

The network produces a final output, $\overline{z}$, which is the element-wise sum of the prediction vectors generated by the modules. Prediction vectors whose elements are near zero (e.g. $\overline{p}_1$, $\overline{p}_2$, $\overline{p}_3$) will contribute little to the sum. Thus, the output of the network will be $\overline{z} \approx \overline{p}_2$, which for the Sentence Completion task, is the correct output, blicket.

As discussed in Section 4.4.1, the grounds for generalization in Phillips' Chair-Colour task are comparable to the grounds for Marcus' Sentence Completion task. The only difference is that the output in Phillips' task is identical to the fifth word of the input case, rather than the second, as in the Sentence Completion case. Using the same graphemic or phonetic
representational scheme that would work for the Sentence Completion task, the relational trace network would be perform as for the Sentence Completion task. The only difference would be that it would be the fifth, rather than the second module, which generated the key prediction vector.

Marcus' Binary Identity task could also be handled under similar representational schemes. If the input is interpreted as a set of symbols, then each digit could be represented by a set of 10 nodes, where (1, 0, 0, 0, ..., 0) represents the digit 0, (0, 1, 0, 0, ..., 0) represents the digit 1, (0, 0, 1, 0, ..., 0) represents the digit 2, etc. Alternatively, these could be considered to be part of a larger symbol (grapheme) set (e.g. of size 26 + 10) which includes not only letters, but digits.

As discussed in Section 4.4.1, the structure of the Binary Identity training cases is simpler than for the other two tasks discussed in this section. In the current task, the output of each training case is identical to its input. Visual cues in Marcus' presentation of the task suggest obvious chunking of the data into four-digit numbers, analogous to words. These could be presented to the relational trace network as in the same way as for the other tasks discussed in this section. The only difference would be that, only a single module would be needed to compare the input of each training case with its output.

Even if the original task were interpreted as having been specified in ground form (i.e. the first training input to the network would be the binary vector 1010, the first training output would be the binary vector 1010, etc.) the relational trace network could generalize appropriately. For example, for the first training case, the two binary vectors to be compared would be 1010 (the input of the training case) and 1010 (the output of the training case). Since any two binary vectors can be compared by a single module, the relational trace vector would be the scalar output, $r$, of this first module. Since in every training case, the input is identical to the output, this output would be zero, consistently throughout the training set. For the test case, 1111 the prediction vector would then be $\bar{p} = \bar{x} = 1111$, since from the training set, $r = 0.$
### Table 5.3: Similar Inputs

<table>
<thead>
<tr>
<th></th>
<th>r</th>
<th>o</th>
<th>s</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Recall that outputs (shown here on the diagonal) are created by an XOR of the inputs.*
### Table 5.4: Dissimilar Inputs\textsuperscript{b}

| i  | 0   | 1   | 0   | 1   | 0   | 1   | 0   | 1   | 0   | 1   | 0   | 1   | 0   | 1   | 0   | 1   | 0   | 1   | 0   | 1   | 0   | 1   | 0   | 1   | 0   | 1   |
|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| s  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 1   | 1   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| r  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| o  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| s  | 1   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| e  | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

\textsuperscript{b} Recall that outputs (shown here on the diagonal) are created by an XOR of the inputs.
5.3 Summary

This chapter proposes ways in which Winner-Take-All networks might play a role in the solution of classes of problems identified in the previous chapter. In particular, a Winner-Take-All architecture was presented which was shown to be able to perform Cross-Dimensional generalization tasks that appear to be beyond the scope of feedforward networks trained by back-propagation. Winner-Take-All networks were shown not to have the property of training independence. Training independence has previously been associated by Marcus with back-propagation networks and their inability to learn across nodes [52, 53, p. 47]. This property may help to explain some of the differences in performance between the two types of networks, since cross-dimensional generalization does appear to involve learning across nodes.

An approach to a second class of generalization problems — Core Relational Generalization — was also introduced in this chapter. An architecture was presented in which Winner-Take-All networks could play a limited but central role. The method requires a number of subnetworks that can compute the XOR function of its inputs. This role could be filled by WTA networks, since they have been previously shown [46, 11] to be able to learn the XOR function in biologically plausible ways. Based on this architecture, the Relational Trace technique illustrates how node activations, rather than weight change, can be used to perform types of generalization tasks that have played a central role in the previous research literature.

Implications of this and previous chapters are discussed in the concluding chapter of the thesis.
Chapter 6

Conclusion

The types of generalization problems that are the subject of this thesis have been of interest to researchers because they highlight apparent differences among the capacities of humans, symbolic algorithms, and standard eliminativist neural networks. Although previous formulations of the research problem might not capture precisely or consistently what is unique about the types of generalization in question, the suggestion there are at least two types of generalization seems to resonate with many researchers.

This thesis set out to examine the foundations of how we formulate the research problem itself, to consider whether an alternative and more unified view of the problem space is possible, and to investigate solutions to problems it raises. In doing so, the thesis makes the following contributions: It reveals important inconsistencies and unstated, hence, unsupported assumptions in previous problem formulations. In particular, the analysis identifies fundamental issues of representation and task form and shows how they complicate the problem of determining task equivalence. To help to make assumptions about representation and task form more explicit, hence accessible to analysis, a notational framework and terminology are introduced for specifying generalization tasks. These are used to support a comparison of the previous formulations of the research problem as well as individual sample tasks. Based on this, a more unified formulation of the problem space is presented. Two key underlying factors — the role of learning and grounds for generalization — are identified which can be used to distinguish between First-Order and Second-Order Generalization. By considering how the particular type and pattern of similarity within a task provides the grounds for generalization, further subclasses of
CHAPTER 6. CONCLUSION

generalization problems are identified. Techniques are proposed to address tasks in two key subclasses: *Cross-Dimensional Generalization* and *Core Relational Generalization*. *Winner-Take-All* networks are shown to be able to perform *Cross-Dimensional* generalization tasks which have been considered to be problematic for feedforward back-propagation networks. The thesis shows that *Winner-Take-All* networks do not exhibit the property of *training independence*\(^1\) (a known property of back-propagation networks [52, 53 p. 47]), which may partially explain the differences in performance of the two types of networks on *Cross-Dimensional* tasks. Finally, a way in which *Winner-Take-All* networks could play a role in the solution of *Core Relational* generalization tasks is outlined. The proposed *Relational Trace* technique uses node activations, rather than weight change, to perform Core Relational generalization.

Previous formulations of the research problem and associated empirical studies, such as the infant study by Marcus et al. [56], have attracted a great deal of interest from researchers. Such works explore questions that lie at the convergence of fields such as Computing Science, Cognitive Science, Artificial Intelligence, Psychology, Linguistics, Philosophy, and Neuroscience. The research presented in this thesis has implications for questions at the centre of this interest. In particular, the Relational Trace network is designed to perform tasks such as Marcus’ Binary Identity and Sentence Completion tasks, as well as Phillips’ Cross-Category and Non-Category Generalization examples\(^2\). These tasks have been put forth as problems which differentiate the abilities of humans from those of currently known eliminativist networks. This invites the question, does the Relational Trace network avoid implementing variables or variable binding? No claim is made that this is the case. Rather, it could be argued that the prediction vectors, \(\overrightarrow{p}\), (see Figure 5.12) act as variables, since each corresponds to a particular position, in which any word could appear. Instances are bound to the variables by

---

\(^1\) Note that, although training independence is an obstacle for Universal Generalization [52, 53 p. 47] and forms of Strong Systematicity [62], a lack of training independence does not necessarily allow a network to solve such problems. No claim is made here that *Winner-Take-All* networks are in themselves sufficient to solve such problems or the types of linking problems described in Sections 4.4.1 (Core Relational Generalization) and 4.4.2 (Composite and Special-Purpose Relations).

\(^2\) While the tasks of the infant grammar study [56] are not in the class which this architecture is designed to perform, the more general strategy of computing the two-dimensional relational trace of a task provides key information about the expected structure of the test cases.
the node activations. For example, the variable: 'the second word of the sentence' would be represented by the nodes of $\mathbf{P}_2$. The instantiation of that variable (e.g. *blicket*) would be bound to the variable by the activation values of the nodes of $\mathbf{P}_2$. While a network that performs Universal Generalization without implementing variables or variable binding would be a startling (and perhaps impossible) result, *implementational* connectionist solutions are by no means insignificant. Phillips [63] and Clark and Thornton [15] observe that the types of generalization they identify are problematic even for classically symbolic algorithms. Strong and Semantic Systematicity also challenge both classically symbolic methods and connectionist networks [12, 38]. Concerning attempts to explain higher cognitive functions even as well as symbolic AI, Hadley observes that "even if a given c-net architecture essentially implements a classical machine, at the computational level, such an implementation would still be vastly important if it revealed how high-level classical processes could be realized in brainlike systems" [34]. Although the techniques presented in this thesis are not purported to be biologically realistic, to the extent that they use Winner-Take-All networks, rather than backpropagating networks, they are a move in the direction of biological plausibility.

While the XOR components of the Relational Trace network would be learnable in a biologically plausible way by Competitive Learning (Winner-Take-All), whether all of the weights of the Relational Trace network could be acquired by learning remains an open problem. It is difficult to see how this might be done using existing algorithms, except perhaps by the somewhat contrived strategy of training of each module individually. Alternatively, one might resort to the unlikely argument that the appropriate weights could be present innately. However, as noted above, the network is not intended to be viewed as a biologically accurate model. Rather, it is an abstract processing model, aspects of which might be reflected in biological networks that solve the same problems.

The Relational Trace method also raises new questions for future research about what kinds of training cases would be relevant. Marcus might argue that only previous training on the *task* (e.g. training on sentences that do not fit the template $A(\mathbf{x})$ is $a(\mathbf{x})$ would be relevant

---

3 Hadley uses the term "c-net" here as an abbreviation for the term "connectionist network" [34].
to the Sentence Prediction task — see Section 3.1.2). The structure of the Relational Trace network suggests that once the network has learned to handle each feature (e.g. letter or phoneme) in each position, it could handle any novel combination of features, such as \textit{A blicket is a blicket}. Thus, sentences that fit the template \textit{A (x) is a (x)} might never need to have been encountered during training. While this situation might satisfy Marcus in that the test case falls outside of the training space of the immediate task, the test cases need not activate any node that was not varied in the set used to train the network. Finally, future research could investigate further how combinations of First and Second-Order techniques might be applied to problems involving Composite and Special-Purpose Relations.

Ultimately, some of the dilemmas currently faced by researchers comparing human and connectionist generalization may be relieved by advances in non-invasive neural imaging technologies. It is hoped that the research presented in this thesis will also prove to be helpful in the effort to understand, appreciate, and enhance intelligence — human or otherwise.
Appendix

Proofs

This Appendix provides supporting details for the arguments presented in Section 5.1.1 of Chapter 5, regarding the Winner-Take-All technique for Cross-Dimensional Generalization. The first two sections of this appendix describes the conditions under which the data clusters in a Cross-Dimensional Generalization tasks are sufficiently distinct to be approached by the proposed Winner-Take-All method. Binary classification tasks are discussed in Section A.1, while tasks that classify inputs into multiple distinct classes are considered in Section A.2. Finally, Section A.3 an analysis of the expected error of the method.

A.1 Binary Classification

The following two sections describe the how the clusters of training data in Cross-Dimensional generalization tasks are separated according to class in the expanded input space. Sections A.1.1 considers methods that use the inner product as a similarity measure. Results for methods that use Euclidean distance measures are given in Section A.1.2.

Although the dimensionality of the (expanded) input space for the general case is higher than for the restricted case discussed in Section 5.1.1 of Chapter 5, and hence, difficult to illustrate, the situation is analogous to the three-dimensional one. By including the class labels as part of each input vector, training cases from distinct classes are mapped to separate (hyper) planes.
A.1.1 Inner Product Similarity Measures

Consider tasks with $l \geq 2$ elements in each formal input vector, where those elements take values in the closed interval $[0, R]$, and where class labels represented by input values of $\pm Q$. For example, in Chapter 5, Table 5.1 b) shows a task for which $l = 2$, $R = 1$ and $Q = 1$.

Let $\overline{a} = (a_1, a_2, ..., a_l)$ and $\overline{b} = (b_1, b_2, ..., b_l)$ be any two distinct training vectors in the original input space that belong to the same class. Then, their corresponding inputs in the expanded input space will be of the form: $\overline{a}' = (a_1, a_2, ..., a_l, a_{l+1})$, $\overline{b}' = (b_1, b_2, ..., b_l, b_{l+1})$. Likewise, let $\overline{c} = (c_1, c_2, ..., c_l)$ and $\overline{d} = (d_1, d_2, ..., d_l)$ be any two distinct training vectors from the original input space that belong to different classes. Then, their corresponding inputs in the expanded input space will be of the form: $\overline{c}' = (c_1, c_2, ..., c_l, c_{l+1})$, $\overline{d}' = (d_1, d_2, ..., d_l, d_{l+1})$.

Let $\alpha$ be the angle between $\overline{a}'$ and $\overline{b}'$ in the expanded input space (where $\overline{a} \neq \overline{b}$ are from the same class). Let $\beta$ be the angle between $\overline{c}'$ and $\overline{d}'$ in the expanded input space (where $\overline{c} \neq \overline{d}$ are from the different classes).

To ensure useful clustering, we need to show that, in the expanded input space, the inner product (i.e. proximity) of two vectors from the same class will always be greater than the inner product of two vectors from different classes.

For inputs that have been normalized (i.e. to a common, constant length, $C$), we have that the inner product of any two vectors, $\overline{x}$ and $\overline{y}$, is $\overline{x} \cdot \overline{y} = |\overline{x}| |\overline{y}| \cos \theta = C^2 \cos \theta$. This means that the inner product is minimal when $\cos \theta$ is minimal. Thus, we want to show that $\cos \alpha > \cos \beta$.

So two vectors $\overline{a}'$ and $\overline{b}'$ from the same class will have the greatest angle of separation, $\alpha$, when their inner product is at a minimum. Since they will have the same class label (i.e. either $a_{l+1} = b_{l+1} = \pm Q$, or $a_{l+1} = b_{l+1} = -Q$), their inner product will be minimal when $\overline{a}$ and $\overline{b}$ are of the form $(R, 0, R, 0, ...)$, and $(0, R, 0, R, ...)$, respectively. That is, no two vectors in the same class can be separated by more than the angle between $\overline{a}' = (R, 0, R, 0, ..., a_{l+1})$ and $\overline{b}' = (0, R, 0, R, ..., b_{l+1})$ — or their normalized counter parts.$^2$

---

$^1$ Note the direction of this inequality. In the examples of Chapter 5, we showed that $\alpha < \beta$. When $\cos \theta$ and the dot product are large, $\theta$ is small, so the comparable result we need is $\cos \alpha > \cos \beta$.
If \( \overline{a} \) and \( \overline{b} \) are from the same class, \( \cos \alpha \) will be at least at least

\[
\cos \alpha = \frac{\overline{a} \cdot \overline{b}'}{|\overline{a}'||\overline{b}|} \geq \frac{1}{\sqrt{\sum_{i=1}^{l} a_i^2 + Q^2}} \frac{1}{\sqrt{\sum_{i=1}^{l} b_i^2 + Q^2}} = \frac{0 + Q^2}{\left( \sqrt{\frac{lR^2}{2} + Q^2} \right)^2} = \frac{Q^2}{\frac{lR^2}{2} + Q^2}
\]

when \( l \) is even. The above inequality also holds for \( l \) odd, since

\[
\cos \alpha = \frac{\overline{a} \cdot \overline{b}'}{|\overline{a}'||\overline{b}|} \geq \frac{Q^2}{\sqrt{\frac{(l+1)R^2}{2} + Q^2} \sqrt{\frac{(l-1)R^2}{2} + Q^2}} > \frac{Q^2}{\frac{lR^2}{2} + Q^2}
\]

So for all \( l \),

\[
\cos \alpha \geq \frac{Q^2}{\frac{lR^2}{2} + Q^2}
\]

For two vectors from the different classes, the proximity will be maximal when \( \overline{c} \) and \( \overline{d} \) are both of the form \((R, R, \ldots, R)\). If \( \overline{c} \) and \( \overline{d} \) are from different classes, \( \cos \beta \) will be at least at most

\[
\cos \beta = \frac{\overline{c}' \cdot \overline{d}}{|\overline{c}'||\overline{d}|} \geq \frac{lR^2 - Q^2}{\left( \sqrt{lR^2 + Q^2} \right)^2} = \frac{lR^2 - Q^2}{lR^2 + Q^2}
\]

2. In this regard, it does not matter whether the vectors are normalized or not. Normalizing a vector does not change its direction, so \( \alpha \) and its cosine would be unchanged.

3. Note that, since \( \overline{c} \) and \( \overline{d} \) are distinct vectors, the inequality here is strict. That is, \( c_i \neq d_i \), for at least one \( i \leq l \).
since the input labels are \( +Q \) and \(-Q \). That is
\[
\cos \beta \geq \frac{lR^2 - Q^2}{lR^2 + Q^2}
\] (A.5)

When \( \cos \alpha > \cos \beta \), the classes will be well-separated. From A.5 and above, we see that this will be the case when
\[
\frac{Q^2}{lR^2 + Q^2} > \frac{lR^2 - Q^2}{lR^2 + Q^2}
\]

which is true when
\[
\frac{2Q^2}{lR^2 + 2Q^2} > \frac{lR^2 - Q^2}{lR^2 + Q^2}
\]
so
\[
2Q^2(lR^2 + Q^2) > (lR^2 - Q^2)(lR^2 + 2Q^2)
\]

Expanding, gives:
\[
2Q^2lR^2 + 2Q^4 > l^2R^4 + 2Q^2lR^2 - Q^2lR^2 - 2Q^4
\]

which reduces to
\[
4Q^4 > l^2R^4 - Q^2lR^2
\]

So the classes will be well-separated in the expanded input space when
\[
4Q^4 + Q^2lR^2 - l^2R^4 > 0
\] (A.6)

For example, when \( Q = 1, R = 1 \), and \( l = 2 \), the classes will be well-separated, since for these values,
\[
4Q^4 + Q^2lR^2 - l^2R^4 = 4 - 2 + 4 = 2 > 0
\]

so the inequality A.6 is satisfied.
One implication of this is that, if the classes are not well-separated for a particular combination of values \( I, R, \) and \( Q \), they can be made so by scaling the values used to represent the input labels. For example, when \( I = 3, R = 1 \) and \( Q = 1 \), the inputs are not well-separated, since for these values, the inequality A.6 above does not hold. But scaling \( Q \) by a factor of 2 would give good separation, since A.6 becomes:

\[
4(2)^4 > I(I - 4)
\]

which is true not only when \( I = 3 \), but for all \( I \leq 10 \). So scaling \( Q \) by a factor of provides good separation for problems whose original inputs space has up to ten dimensions.

Scaling the labels upward has the effect of compressing the remaining elements of the expanded input vectors into a smaller range. So in practice, \( Q \) should not be scaled much more than necessary to satisfy the relevant inequality, since realistic networks would have limitations on the precision they could achieve.

### A.1.2 Euclidean Distance Measures

This section gives separability results for competitive learning methods that avoid the need for normalizing inputs by using Euclidean distance measures [17, 18]. Consider tasks with \( I \) elements in each formal input vector, with formal inputs in the range \([0, R]\) and class labels represented by input values of \( \pm Q \).

For \( I \) formal inputs, let \( \overline{a} = (a_1, a_2, \ldots, a_I) \neq \overline{b} = (b_1, b_2, \ldots, b_I) \) be any two distinct vectors in the original input space. Then, their corresponding inputs in the expanded input space will be of the form: \( \overline{a}' = (a_1, a_2, \ldots, a_I, y) \), \( \overline{b}' = (b_1, b_2, \ldots, b_I, y) \). Consider formal input vectors whose elements have values in the range \([0..1]\). If \( \overline{a} \) and \( \overline{b} \) are from different classes, \( \overline{a}' \) and \( \overline{b}' \) will be separated from each other by a distance of \textit{at least}\(^4\)

\[
|\overline{a}' - \overline{b}'| = \sqrt{\sum_{i=1}^{I} (a_i - b_i)^2 + 2^2} \geq \sqrt{0 + 4} = 2
\]

---

\(^4\) Note that, since \( \overline{a} \) and \( \overline{b} \) are distinct vectors, the inequality here is strict. That is, \( a_i \neq b_i \), for at least one \( i \leq I \).
However, for input vectors \( \mathbf{c} \neq \mathbf{d} \) in the same class, \( \mathbf{c}' \) and \( \mathbf{d}' \) will be separated from each other by a distance of at most

\[
|\mathbf{c}' - \mathbf{d}'| = \sqrt{\sum_{i=1}^{l} (c_i - d_i)^2 + 0} \leq \sqrt{\sum_{i=1}^{l} 1} = \sqrt{l}
\]

So for \( l \leq 4 \), the classes will be well-separated, since

\[
|\mathbf{c}' - \mathbf{d}'| \leq \sqrt{l} \leq 2 < |\mathbf{a}' - \mathbf{b}'|
\]

That is in the expanded input space, not only do the two classes not overlap, but even their outermost edges are separated by a distance that is at least as large as the maximum spread of each class, when the formal input space has fewer than five dimensions.

More generally, for all \( l \geq 2 \), in tasks with formal inputs in the range \([0, R]\) and class labels represented by input values of \( \pm Q \), if \( \mathbf{a} \) and \( \mathbf{b} \) are from different classes, \( \mathbf{a}' \) and \( \mathbf{b}' \) will be separated from each other by a distance of at least

\[
|\mathbf{a}' - \mathbf{b}'| = \sqrt{\sum_{i=1}^{l} (a_i - b_i)^2 + (2Q)^2} > \sqrt{0 + 4Q^2} = 2Q
\]

Whereas for input vectors \( \mathbf{c} \neq \mathbf{d} \) in the same class, \( \mathbf{c}' \) and \( \mathbf{d}' \) will be separated from each other by a distance of at most

\[
|\mathbf{c}' - \mathbf{d}'| = \sqrt{\sum_{i=1}^{l} (c_i - d_i)^2 + 0} \leq \sqrt{\sum_{i=1}^{l} R^2} = R\sqrt{l}
\]
APPENDIX. PROOFS

So for all \( l > 0 \), the classes will be well-separated, when

\[
| \tilde{e}' - \tilde{a}' | \leq R\sqrt{l} \leq Q2 < | \tilde{a}' - \tilde{b}' |
\]

So when

\[
Q \geq \frac{R\sqrt{l}}{2}
\]

the two classes will be separated by a distance that is at least as large as the maximum spread of each class, for any number of input dimensions.

In tasks with \( l > 0 \), formal inputs in the range \([0, R]\), and class labels represented by input values of \( \pm Q \), the inputs to the WTA(2) output layer of the network will also be well-separated according to class, for the following reasons.

Two training cases that belong to the same class will have the same class label, \( x_{j+1} \). If they happen to be nearest to the same centroid, then by the end of training they will activate the same hidden node, and so provide identical vectors to the WTA(2) layer. So the inputs, \( \tilde{e} = \tilde{f} \), to the WTA(2) output layer will have a distance of \( | \tilde{e} - \tilde{f} | = 0 \).

Otherwise, if they are to nearest to two different centroids, they will activate two different hidden nodes. So the inputs, \( \tilde{e} \) and \( \tilde{f} \), to the WTA(2) layer will differ by 1 in exactly 2 places. This gives a distance of \( | \tilde{e} - \tilde{f} | = \sqrt{2} \).

In the input space to the WTA(2) layer, vectors, \( \tilde{g} \) and \( \tilde{h} \), for inputs from different classes will be separated by a distance of

\[
| \tilde{g} - \tilde{h} | = \sqrt{1^2 + 1^2 + (2Q)^2} = \sqrt{2 + 4Q^2}
\]

since the output vectors from the hidden layer will differ by one in exactly two places, and the class labels will differ by \( 2Q \).

Thus, in the input space to the WTA(2) layer,

\[
| \tilde{e} - \tilde{j} | \leq \sqrt{2} < \sqrt{2 + 4Q^2} = | \tilde{g} - \tilde{h} |
\]
so the distance between training vectors from the same class will be less than the distance between training vectors from the different classes, for all \( i, R, \) and \( Q > 0 \).

## A.2 Multiple Classes

The following section describes the separation properties of the data for tasks that classify training cases into multiple distinct classes.

### A.2.1 Inner Product Similarity Measures

Consider tasks with \( l > 2 \) elements in each formal input vector, with formal inputs in the range \([0, R]\) and class labels represented by input values of \( Q \). To represent labels for \( m \) distinct classes, the input labels would be of the form \((y_1, y_2, \ldots, y_m)\), where \( y_i = Q \) when that training case belongs to class \( i; y_i = 0 \), otherwise. Task c) shows how this would appear for \( m = 2 \).

Let \( \vec{a} = (a_1, a_2, \ldots, a_l) \neq \vec{b} = (b_1, b_2, \ldots, b_l) \) be any two distinct vectors from the original input space that belong to the same class. Then, their corresponding inputs in the expanded input space will be of the form: \( \vec{a}' = (a_1, a_2, \ldots, a_l, y_1, y_2, \ldots, y_m) \), \( \vec{b}' = (b_1, b_2, \ldots, b_l, y_1, y_2, \ldots, y_m) \).

For normalized inputs, the inner product of two vectors is \( \vec{x} \cdot \vec{y} = \frac{|x|}{|y|} \cos \theta = C \cos \theta \), for some constant, \( C \). This means that the inner product (i.e. proximity) is minimal when \( \cos \theta \) is minimal. So for two vectors from the same class, this will be minimal when \( \vec{a} \) and \( \vec{b} \) are of the form \((R, 0, R, 0, \ldots)\), and \((0, R, 0, R, \ldots)\), respectively. Since the classes are distinct, no training case would be labelled as belonging to more than one class, so the label portion, \((y_1, y_2, \ldots, y_m)\), of the training inputs will have exactly one element \( = Q \). The remaining \( y_i = 0 \).

Let \( \alpha \) be the angle between \( \vec{a}' \) and \( \vec{b}' \) in the expanded input space. If \( \vec{a} \) and \( \vec{b} \) are from the same class, then for all \( m \), \( \cos \alpha \) will be at least \( \cos \alpha = \frac{\vec{a}' \cdot \vec{b}'}{|a||b|} \geq \frac{Q^2}{LR^2 + Q^2} \).
For two vectors from the different classes, the proximity will be maximal when \( \tilde{c} \) and \( \tilde{d} \) are both of the form \((R, R, \ldots, R)\). Let \( \beta \) be the angle between \( \tilde{c}' \) and \( \tilde{d}' \) in the expanded input space. If \( \tilde{c} \) and \( \tilde{d} \) are from different classes, \( \cos \beta \) will be at least at most \(^5\)

\[
\cos \beta = \frac{\tilde{c}' \cdot \tilde{d}'}{|\tilde{c}'||\tilde{d}'|} \geq \frac{lR^2 - 0}{(\sqrt{1R^2 + Q^2})^2} = \frac{lR^2}{lR^2 + Q^2}
\]

When \( \cos \alpha > \cos \beta \), the classes will be well-separated. This will be the case when

\[
\frac{Q^2}{lR^2 + Q^2} > \frac{lR^2}{lR^2 + Q^2}
\]

which is true when

\[
\frac{2Q^2}{lR^2 + 2Q^2} > \frac{lR^2}{lR^2 + Q^2}
\]

and

\[
2Q^2(lR^2 + Q^2) > lR^2(lR^2 + 2Q^2)
\]

Expanding, gives:

\[
2Q^2lR^2 + 2Q^4 > l^2R^4 + 2Q^2lR^2
\]

So the classes will be well-separated in the expanded input space when

\[
2Q^4 > l^2R^4
\]

\(^5\) Note that, since \( \tilde{c} \) and \( \tilde{d} \) are distinct vectors, the inequality here is strict. That is, \( c_i \neq d_i \), for at least one \( i \leq l \).
which is

\[ Q^2 > \frac{IR^2}{\sqrt{2}} \]  

So, for example, when \( Q = 2, R = 1 \), the classes will be well-separated when \( I \leq 5 \), since

\[ 4\sqrt{2} > l \]

### A.2.2 Euclidean Distance Measures

Using Euclidean distance as the measure of similarity, we want the distance between vectors from different classes to be greater than between vectors in the same class (in the expanded input space). Let \( \vec{a} = (a_1, a_2, \ldots, a_I) \neq \vec{b} = (b_1, b_2, \ldots, b_I) \) be any two distinct vectors from the original input space. In the expanded input space, these will be of the form: \( \vec{a}' = (a_1, a_2, \ldots, a_I, a_{I+1}, a_{I+2}, \ldots, a_{I+M}) \), \( \vec{b}' = (b_1, b_2, \ldots, b_I, b_{I+1}, b_{I+2}, \ldots, b_{I+M}) \).

When \( \vec{a} \) and \( \vec{b} \) are from different classes, the distance between \( \vec{a}' \) and \( \vec{b}' \) will be

\[
|\vec{a}' - \vec{b}'| = \sqrt{\sum_{i=1}^{I} (a_i - b_i)^2 + 2Q^2} > Q\sqrt{2}
\]

since the label portion of each will differ (by \( Q \)) in exactly two places. For example, if they belong to class 1 and class 2, respectively, then \( \vec{a}' = (a_1, a_2, \ldots, a_I, Q, 0, 0, \ldots, 0) \) and \( \vec{b}' = (b_1, b_2, \ldots, b_I, Q, 0, 0, \ldots, 0) \).

For two distinct input vectors, \( \vec{c} \neq \vec{d} \), that are from the same class, the distance between \( \vec{c}' \) and \( \vec{d}' \) will be

\[
|\vec{c}' - \vec{d}'| = \sqrt{\sum_{i=1}^{I} (c_i - d_i)^2 + 0} \leq \sqrt{IR^2 + 0} = R\sqrt{l}
\]

since the label portion of each will be identical. For example, if they both belong to class 2, then \( \vec{c}' = (c_1, c_2, \ldots, c_I, Q, 0, 0, \ldots, 0) \) and \( \vec{d}' = (d_1, d_2, \ldots, d_I, Q, 0, 0, \ldots, 0) \). Thus, input vectors...
from *different* classes will be more distant from each other in the expanded input space than vectors from the *same* class when

\[ Q > R \frac{1}{N^2} \]

### A.3 Approximation of Training Data

This section undertakes an analysis of the expected generalization performance of the WTA technique introduced in Section 5.1.1 of Chapter 5 when the network is tested on the known, but now unlabelled, training cases.

In tasks where the GSI is near zero, very few training items would have an item with the *same* class label as its nearest neighbour. So there would be very few, if any, uniformly labelled subclusters in the original input space. To perform well when tested on the unlabelled training cases, the number of nodes in the hidden layer would have to be equal to the number of training cases (so that each node finds a subcluster of size 1). Unlabelled cases from the training set would then be classified according to their nearest centroid (as measured in what is effectively the original input space). Since clusters are of size 1, the nearest centroid is the training case itself. So the label provided by the network would be the same as the label it had during training. This, of necessity, provides more of a look-up table than a generalization method.

At the other extreme, if the GSI is near 1, then even in the original training space, the data would cluster well according to class. That is, very few, if any, training items would have a nearest neighbour that has a *different* class label. So most or all clusters in the original input space would be uniformly labelled and there would be little or no overlap between clusters from different classes. To find the centroids of \( n \) such clusters, a hidden layer of \( n \) nodes would be needed. When tested on an unlabelled training case, the network will assign the label of cluster whose centroid is nearest in the original input space. This will be the cluster to which the item belongs in the original input space. Since the overlap between clusters in the original
input space is negligible, the label of this centroid will tend to be very consistent with the label of that case had during training.

For problems with intermediate GSI values, there would likely be some uniformly-labelled subclusters in the original input space. Recall that if, in the expanded input space, there are \( p \geq 2 \) well-separated clusters in each of \( m \) distinct classes, a network with \( n=m*p \) hidden nodes can find their centroids. First, let us consider binary classification problems (i.e. where \( m=2 \)) and where the formal input vectors have \( l=2 \) elements.

In the worst case (i.e. the coarsest approximation of the training data) the subclusters in each class plane will be as large as possible. Let \( R \) be the largest cluster radius that would allow \( p>1 \) clusters to be well-separated in a class plane. The since the subclusters found within a given class plane must be well-separated\(^6\), each pair must be separated by a distance that is somewhat greater than the radius of the largest of the two subclusters. This is illustrated in Figure A.1, where the shaded discs indicate the subclusters in a particular class plane. The white discs indicate their minimum separation.

In the original input space — which is where proximity to cluster centroids is evaluated during testing — the clusters of training cases from different classes may, in general, have some overlap. In the worst cases situation shown in Figure A.1, some overlap is relatively likely. In the following analysis, the expected error in classification of unlabelled training cases is estimated for this worst case.

Figure A.2 a) to d) show examples of the ways in which clusters from distinct classes may be positioned in relation to each other in the original input space. The circles in Figure A.2 a) to d) show the boundaries of the clusters of uniformly-labelled training cases\(^7\). The heavy vertical line in each of these cases shows the learned boundary which will be used to classify novel test cases. All points on this line are equidistant from the centroids of two clusters.

---

\(^6\) Recall that only tasks which satisfy this condition (as required by Rumelhart and Zipser [67]) would guarantee that the network converges during training.

\(^7\) Recall that the clusters must be uniformly-labelled, since they were found during training in the expanded input space, in which classes are very well-separated.
Figure A.1: Minimum Separation of Clusters in a Class Plane

Figure A.2: Overlap of Clusters of Training Items

\( d = 2R \)  
Class A \hspace{1cm} Class B  
a)

\( R < d < 2R \)  
Class A \hspace{1cm} Class B  
b)

\( 0 < d < R \)  
Class A \hspace{1cm} Class B  
c)

\( d \approx 0 \)  
Class A \hspace{1cm} Class B  
d)
In Figure A.2 suppose that the training cases in the cluster on the left of each of figures a) to d) contains training cases from Class A, and the cluster on the right of each figure contains training cases from Class B. In Figure A.2 b), all test cases that fall to the left of the heavy vertical line will be classified as belonging to Class A, whereas all those which fall to the right that line will be classified as belonging to Class B. So, if there are any training cases from Class A that fall within the shaded area to the right of the learned boundary, then these would be misclassified (when presented as unlabelled test cases), since they are nearest to the centroid of a cluster which, during training, contained only cases labelled Class B. Likewise, unlabelled training cases from Class B that fall to the left of the heavy vertical line would be misclassified, during testing, as belonging to Class A.

The error rate introduced by two clusters that are separated by a distance \( d \) will thus be the number of training cases that fall within the shaded area, divided by the number of training cases that fall within either of the two discs (i.e. which belong to either of the two clusters). Let \( s \) be the number of training cases in each cluster. Then the number of training cases in two clusters is \( 2s \). If we assume that, in the general case, a training case is equally likely to appear at any position within its disc, then the number of training cases in the shaded region would be (number of training cases in two clusters) * (area of shaded region) / (area of two discs):

\[
\text{NumErrors} = \frac{2s(A)}{2\pi R^2}
\]

Dividing by the number of training items in the two clusters, gives the error rate:

\[
\text{ErrorRate} = \frac{2s(A)}{2s(2\pi R^2)} = \frac{A}{2\pi R^2}
\]

where the area, \( A \), of the shaded region [79] is

\[
A = 2R^2 \arccos \left( \frac{d}{2R} \right) - \frac{d}{2} \sqrt{4R^2 - d^2} \quad (A.10)
\]

---

8 Recall that each of the \( n \) clusters found by competitive learning contain approximately the same number of training cases [67].
9 In this notation, \( \arccos(x) = \arccos(x) = \cos^{-1}(x) \).
When $d = 2R$, there is no overlap between the boundaries of clusters, so no training items will be misclassified (see Figure A.2 a). As the distance, $d$, between the centroids of the clusters decreases, the proportion of training items in each subcluster that will be mislabelled during testing increases (see Figure A.2 c, d). As $d$ approaches zero, the error rate, approaches 50% (see Figure A.2 e).

The expected value [47, p. 259] of the error rate can be determined by integrating over all possible locations of the second centroid relative to the first:

$$E(\text{Error Rate}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} ErrorRate(r, \theta)f(r, \theta)rdrd\theta \quad (A.11)$$

where $f(r, \theta)$ is the joint probability density function [47, p. 44]:

$$f(r, \theta) = \begin{cases} \frac{1}{\pi(2R)^2} & 0 \leq r \leq 2R \\ 0 & \text{otherwise} \end{cases} \quad (A.12)$$

That is, the second centroid is equally likely to appear at any position\(^{10}\) within the circle of radius $2R$ from the first centroid (see Figure A.1).

Combining A.10, A.11, and A.12 (and expressing the Error Rate as a function of $r$, rather than $d$) gives:

$$E(\text{Error Rate}) = \int_{0}^{2\pi} \int_{0}^{2R} \left( \frac{2R^2 \cos\left(\frac{r}{2R}\right) - \frac{r}{2} \sqrt{4R^2 - r^2}}{2\pi R^2} \right) \frac{1}{\pi(2R)^2} rdrd\theta \quad (A.13)$$

\(^{10}\)A few cases could have $d > 2R$ (see the black region of Figure A.1). These are taken into account later in this section. In addition, this approximation ignores boundary conditions (i.e. where subclusters would be very near the border of a class plane).
Factoring out constants and integrating over $d\theta$ and gives

$$E(\text{ErrorRate}) = \frac{2\pi}{(2\pi R^2)^2}\left(\frac{4R^2}{2}\right)^2 \int_0^a r \cos\left(\frac{r}{a}\right) - \frac{r^2}{a^2\sqrt{a^2 - r^2}} dr$$  \hspace{1cm} (A.14)

If we set $a = 2R$, this reduces to

$$E(\text{ErrorRate}) = \frac{1}{2\pi R^2} \int_0^a r \cos\left(\frac{r}{a}\right) - \frac{r^2}{a^2\sqrt{a^2 - r^2}} dr$$  \hspace{1cm} (A.15)

We can solve separately for $B$, and $C$

$$E(\text{ErrorRate}) = \frac{1}{2\pi R^2} \left[ \int_0^a \frac{r \cos\left(\frac{r}{a}\right)}{a} dr - \frac{1}{a^2} \int_0^a r^2 \sqrt{a^2 - r^2} dr \right]$$  \hspace{1cm} (A.16)

From a standard table of integrals [58, pp. 522-527], $C$ evaluates to

$$C = \frac{1}{a^2} \int_0^a r^2 \sqrt{a^2 - r^2} dr$$  \hspace{1cm} (A.17)

$$= \frac{1}{a^2} \left[ (2r^2 - a^2)\sqrt{a^2 - r^2} + a^4 \sin^{-1}\left(\frac{r}{a}\right) \right]_0^a$$

$$= \frac{1}{8a^2} \left[ (a^2\sqrt{a^2 - a^2} + a^4 \sin^{-1}1) - (0) \right]$$

$$= \frac{1}{8a^2} \left[ a^2\pi - 0 \right]$$

$$= \frac{\pi a^2}{16}$$

$$= \frac{\pi R^2}{4}$$

since we had set $a = 2R$. 
To evaluate B, we use integration by parts [27, p. 422]

\[ \int u dv = uv - \int v du \]

with

\[ u = \cos \frac{r}{a}, \quad dv = r dr \]

Then, since from a standard table of derivatives and the chain rule [58, pp. 521], we have

\[ \frac{d}{dr} \left( \cos \frac{r}{a} \right) = \left( \frac{-1}{\sqrt{1 - \left( \frac{r}{a} \right)^2}} \right) \frac{d}{dr} \left( \frac{r}{a} \right) \]

\[ = \left( \frac{-1}{\sqrt{a^2 - r^2}} \right) \frac{1}{a} \]

\[ = \frac{-1}{\sqrt{a^2 - r^2}} \]

\[ du = \frac{-1}{\sqrt{a^2 - r^2}} dr, \quad v = \frac{r^2}{2} \]

So from A.18, A.19, and A.20,

\[ \int r \cos \left( \frac{r}{a} \right) dr = \int u dv = uv - \int v du \]

\[ = \left( \frac{r^2}{2} \cos \frac{r}{a} \right) - \int \frac{-r^2}{2 \sqrt{a^2 - r^2}} dr \]

\[ = \frac{1}{2} \left[ \frac{r^2}{2} \cos \frac{r}{a} + \int \frac{r^2}{\sqrt{a^2 - r^2}} dr \right] \]
From a standard table of integrals [58, pp. 522-527]

\[ \int \frac{r^2}{\sqrt{a^2 - r^2}} dr = -\frac{r}{2} \sqrt{a^2 - r^2} + \frac{a^2}{2} \arcsin \frac{r}{a} \quad (\text{A.22}) \]

Combining A.21 and A.22 gives

\[ \int r \arccos \left( \frac{r}{a} \right) dr = \frac{1}{2} \left[ r^2 \arccos \frac{r}{a} - \frac{r}{2} \sqrt{a^2 - r^2} + \frac{a^2}{2} \arcsin \frac{r}{a} \right] \]

So we can evaluate \( B \) as

\[
B = \int_0^a r \arccos \left( \frac{r}{a} \right) dr \\
= \frac{1}{2} \left[ r^2 \arccos \frac{r}{a} - \frac{r}{2} \sqrt{a^2 - r^2} + \frac{a^2}{2} \arcsin \frac{r}{a} \right]_0^a \\
= \frac{1}{2} \left[ \left( \frac{a^2}{2} \arccos \frac{a}{a} - \frac{a}{2} \sqrt{a^2 - a^2} + \frac{a^2}{2} \arcsin \frac{a}{a} \right) - \left( 0 \arccos \frac{0}{a} - \frac{0}{2} \sqrt{a^2 - 0^2} + \frac{a^2}{2} \arcsin \frac{0}{a} \right) \right] \\
= \frac{1}{2} \left[ \left( 0 - 0 + \frac{a^2}{2} \left( \frac{\pi}{2} \right) \right) - (0) \right] \\
= \frac{\pi a^2}{8} \\
= \frac{\pi (2R)^2}{8} \\
= \frac{\pi R^2}{2} 
\]

since we had set \( a = 2R \).
Substituting this and A.17 into A.16 gives an expected error rate of

\[
E(\text{Error Rate}) = \frac{1}{2\pi R^2} \left[ \int_0^\rho r \cos \left( \frac{r}{a} \right) dr \right] - \frac{1}{a^2} \int_0^a r^2 \sqrt{a^2 - r^2} dr
\]

\[
= \frac{1}{2\pi R^2} \left[ \frac{\pi R^2}{2} - \frac{\pi R^2}{4} \right]
\]

\[
= \frac{1}{2\pi R^2} \left[ \frac{\pi R^2}{4} \right]
\]

\[
= \frac{1}{8}
\]

\[
= 0.125
\]

A few cases could have \( d > 2R \) (see the black region of Figure A.1). These are not taken into account by the calculations described above, but since clusters centered here could not overlap any other clusters, they would not introduce additional error. The integration accounts for \( \eta = \pi/(2\sqrt{3}) = 0.9068997 \approx 90.7\% \) of cases, where \( \eta = \pi/(2\sqrt{3}) \) is the hexagonal packing density for circles [71, p. 202]. So about 9.3\% of cases will have \( d > 2R \). Taking these cases into account gives an expected error rate of \( 0.907 \times (12.5\%) + 0.093 \times (0\%) \approx 11.3\% \).

So even in the worst case described above (as in Figure A.1), we can expect fewer than about 11.3\% of training cases to be misclassified when presented without labels during testing. In higher-dimensional cases (i.e. when there are more than \( l = 2 \) formal inputs), we would expect an even lower error rate, since the density of the subclusters (and hence the probability of overlap of subclusters) would decrease exponentially as \( l \) increases. For \( l = 2 \), the density of the clusters can be estimated by the proportion of the area of a class plane that is occupied by the clusters (see Figure A.1):

\[
\delta \approx \frac{\pi R^2}{\pi (2R)^2} = \frac{1}{4} = 0.25
\]
In general, in $\mathbb{R}^n$, the volume of a hypersphere of radius $R$ is

$$V_n = \frac{S_n R^n}{n}$$

where $S_n$ is the volume of the unit hypersphere in $\mathbb{R}^n$ [80]. So the density of clusters in $\mathbb{R}^l$ is

$$\delta \approx \frac{\left(\frac{S_l R^l}{l}\right)^l}{\left(\frac{S_l(2R)^l}{l}\right)^l} = \frac{lS_l R^l}{lS_l(2R)^l} = \frac{1}{2^l} \quad (A.23)$$

While computing particular values for the expected error of hyperspherical clusters for each dimensionality, $\mathbb{R}^l$, would be mathematically quite complex, we can observe the effect of this exponential reduction of density by calculating the expected error for $l = 3$. For $l = 3$, the region (i.e. volume) of overlap [81] of two clusters (i.e. spheres) is

$$A = \frac{\pi}{12} (4R + d)(2R - d)^2$$

Since the volume of a sphere is $4\pi R^3/3$, we have

$$ErrorRate = \left(\frac{\pi}{12} \left(\frac{4R + r}{2R - r}\right)^2\right) \quad (A.24)$$

Assuming that the second centroid is equally likely to appear at any position within the sphere of radius $2R$ from the first centroid\(^{11}\), we have the joint probability density function $f(r, \theta, \phi)$ [47, p. 44]:

\(^{11}\) As in the corresponding analysis for the two-dimensional case undertaken previously in this section, this approximation ignores boundary conditions (i.e. where subclusters would be very near the border of a class hyperplane).
So the expected value for the Error Rate is [47, p. 259]:

\[ E(\text{Error Rate}) = \int_V \text{Error Rate}(r, \theta, \phi) f(r, \theta, \phi) dV \]  

(A.26)

Substituting into A.26 from A.24 and A.25 gives

\[
E(\text{Error Rate}) = \int_V \left( \frac{\pi}{12} \frac{(4R + r)(2R - r)^2}{2(\frac{4}{3} \pi R^3)} \right) \frac{1}{\frac{4}{3} \pi (2R)^3} dV
\]

Summing over volume elements \(dV = 4\pi r^2 dr\) (which describe a thin spherical shell in spherical coordinates [57, p. 295]) from 0 to \(2R\), we have

\[
E(\text{Error Rate}) = \int_0^{2R} \left( \frac{\pi}{12} \frac{(4R + r)(2R - r)^2}{2(\frac{4}{3} \pi R^3)} \right) \frac{1}{\frac{4}{3} \pi (2R)^3} 4\pi r^2 dr
\]

Simplifying gives

\[
E(\text{Error Rate}) = \frac{3}{4a^6} \int_0^a (2a + r)(a - r)^2 r^2 dr
\]

where \(a = 2R\).

Expanding the integrand, we have

\[
E(\text{Error Rate}) = \frac{3}{4a^6} \int_0^a (2a^3 r^2 - 3a^2 r^3 + r^5) dr
\]
Integrating gives

\[ E(\text{ErrorRate}) = \frac{3}{4a^6} \left( \frac{r^3 2a^3}{3} - \frac{r^4 3a^2}{4} + \frac{r^6}{6} \right) \bigg|_{r=0} \]

which evaluates to

\[ E(\text{ErrorRate}) = \frac{3}{4a^6} \left[ \left( \frac{2a^6}{3} - \frac{3a^6}{4} + a^6 \right) - (0) \right] \]

\[ = \frac{3a^6}{4a^6} \left( \frac{1}{12} \right) \]

\[ = \frac{1}{16} \]

\[ = 0.0625 \]

Since the maximum packing density for spheres is \( \eta \approx 0.78 \) [40], the above integration accounts for approximately 78% of cases. So about 22% of cases will have \( d > 2R \) (and hence, no overlap). Taking these cases into account gives an expected error rate of \( 0.78 \times (6.25\%) + 22\times (0\%) \approx 4.9\% \)

Thus, the expected error rate for problems with \( l = 3 \) formal inputs is approximately 4.9%. So the expected proportion of training cases misclassified in problems with \( l = 3 \) formal inputs (approximately 4.9%) would be less than for the 2-dimensional case (approximately 11.3%). This is consistent with what we might expect from A.23: since the density of cases diminishes exponentially with \( l \), the likelihood of overlap of clusters, and hence the expected error rate diminishes as \( l \) increases.

For tasks that classify data into \( m > 2 \) distinct classes, overlap between clusters is more likely than for the binary case, since each cluster may overlap with up to \( m - 1 \) other clusters (i.e. one from each of the remaining classes). Computing the expected overlap of hyperspherical clusters for each value of \( m \) in each dimensionality, \( \mathcal{R}^l \), would be mathematically quite complex, but in general, we can expect that the best results would be obtained when the data set is
relatively sparse. In tasks where \( m \) is very large relative to \( L \), the data set would likely be quite dense, and there could be considerable overlap between clusters in the original input space. The GSI of such a task would be very low, and so, as for binary classification problems with low GSI discussed above, a cluster size of one would likely give the best approximation that could be provided.
Bibliography


BIBLIOGRAPHY


