

UC Irvine

ICS Technical Reports

Title

Knowledge acquisition via incremental conceptual clustering

Permalink

<https://escholarship.org/uc/item/6df7w73d>

Author

Fisher, Douglas Hayes, Jr.

Publication Date

1987

Peer reviewed

Notice: This Material
may be protected
by Copyright Law
(Title 17 U.S.C.)

Archives
Z
699
C3
no. 87-22
C.2

University of California
Irvine

**Knowledge Acquisition Via
Incremental Conceptual Clustering**

(Technical Report 87-22)

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Information and Computer Science

by

Douglas Hayes Fisher Jr.

Committee in charge:

Professor Dennis Kibler, Chair

Professor Pat Langley

Professor Richard Granger

1987

Copyright © 1987
Douglas Hayes Fisher Jr.
All Rights Reserved

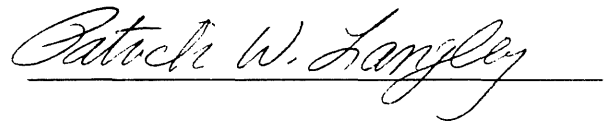
© 1987

Douglas Hayes Fisher Jr.

ALL RIGHTS RESERVED

The dissertation of Douglas Hayes Fisher, Jr. is approved,
and is acceptable in quality and form for
publication on microfilm:







Committee Chair

University of California, Irvine

1987

Dedication

The dissertation is dedicated to my wonderful family: my parents Mary Ellen and Doug, my brother Bruce, and my sisters Diana and Linda (+ Russ, Russ Jr., and Ryan). I am indebted to them all for their love, support, and prodding. Most especially though, it is my extraordinary wife Pat to whom I dedicate my efforts; without her I never would have made it past page one. Oh, I almost forgot my equally wonderful in-laws who lovingly adopted me as a McMahan.

Contents

List of Tables	<i>vii</i>
List of Figures	<i>ix</i>
Acknowledgements	<i>xiii</i>
Curriculum Vitae	<i>xiv</i>
Abstract	<i>xvi</i>
Chapter 1: Introduction	1
1.1 Contributions of the Dissertation	1
1.2 Conceptual Clustering	2
1.3 Basic Level and Typicality Effects	5
1.4 An Overview of Computational and Psychological Antecedents	7
1.5 Methodological Biases	7
1.6 Overview of the Dissertation	17
Chapter 2: Concept Induction in Artificial Intelligence	20
2.1 Chapter Overview	20
2.2 Object and Concept Representation	21
2.3 Learning From Examples	27
2.4 Conceptual Clustering	33
2.5 Incremental Concept Induction	51
2.6 A Performance Task for Conceptual Clustering	59
2.7 Chapter Summary	62
Chapter 3: Psychological Constraints on Concept Induction	64
3.1 Chapter Overview	64
3.2 Typicality Effects and Probabilistic Concepts	65
3.3 Basic Level Effects and Concept Hierarchies	88
3.4 Chapter Summary	100

Chapter 4: COBWEB: An Incremental Conceptual Clustering System. . . .	102
4.1 Chapter Overview	102
4.2 Environment and Knowledge Base	103
4.3 COBWEB's Basic Control Structure	106
4.4 Operator Interactions	118
4.5 Dealing with Missing Information.	126
4.6 Learning Superordinate Concepts	128
4.7 Comparison with Other Systems	133
4.8 Chapter Summary	139
Chapter 5: Classification and Inference	141
5.1 Chapter Overview	141
5.2 Examples of COBWEB Classification Trees.	142
5.3 Default and Normative Values	147
5.4 COBWEB Classification Trees and Inference.	161
5.5 Inference Using Norms.	171
5.6 Conceptual Clustering and Learning from Examples	179
5.7 Induction and Deduction	183
5.8 Chapter Summary	186
Chapter 6: COBWEB as an Incremental Learner.	189
6.1 Chapter Overview	189
6.2 Cost of Assimilating a Single Object	191
6.3 The Quality of Classification Trees	193
6.4 Number of Objects Required For Convergence	197
6.5 Chapter Summary	201
Chapter 7: A Computational Account of Basic Level and Typicality Effects	202
7.1 Chapter Overview	202
7.2 An Indexing Scheme Based On Category Utility.	203

7.3 Basic Level Effects	212
7.4 Typicality Effects	222
7.5 Basic Level and Typicality Effects	249
7.6 Caveats	256
7.7 Chapter Summary	258
Chapter 8: COBWEB/2: Incremental Update of Indexed Memory	260
8.1 Chapter Overview	260
8.2 The COBWEB/2 Algorithm	261
8.3 COBWEB/2 as an Incremental Learner	271
8.4 COBWEB/2 Classification Trees	282
8.5 The Utility of COBWEB/2 Classification Trees for Inference	284
8.6 Chapter Summary	285
Chapter 9: Conclusions	287
9.1 Contributions of the Dissertation	287
9.2 Future Work	290
9.3 Closing Remarks	296
References	298
Appendix A: Data for Computer Experiments	309

List of Tables

Table	Page
1. Sample attributes and domains	22
2. Objects represented as sets of attribute-value pairs	23
3. Concepts of increasing generality	25
4. Linearly separable and non-linearly separable classes	85
5. Animals described in terms of attribute - value pairs	103
6. Expansion of the "animals" node	105
7. An expansion of the "mammal" node	106
8. The control structure of COBWEB	117
9. Pseudo-code for uselessness	124
10. Pseudo-code for promotion	125
11. Pseudo-code for COBWEB with the promotion operator	126
12. A document high-level and attribute-value description	143
13. A sample of senate voting records	145
14. Some typical values of 2 clusters from congressional hierarchy	146
15. Normative values for congressional classes	158
16. Sample soybean disease cases	162
17. Sample thyroid disease cases	168
18. Averaged inference results in the soybean domain	176
19. Averaged inference results in the thyroid domain	178
20. Averages from the ID3 experiments	181
21. An attribute - value encoding of the Murphy and Smith tools	214
22. Attribute value encodings of trees used by Hoffman and Ziessler	219
23. Categories used to test the effect of intra-class similarity on typicality	223
24. Human and computer model response times (1)	228

25.	Categories used to test the effect of inter-class similarity on typicality	231
26.	Human and computer model response times (2)	232
27.	Human and computer model response times (3)	236
28.	Artificial categories used in typicality studies by Rosch and Mervis .	239
29.	Within- and between- category overlap are simultaneously varied . .	242
30.	COBWEB/2 Pseudocode that assumes classification is only operator	263
31.	Consider creating a class and do so if appropriate.	265
32.	Consider merging best hosts and do so if appropriate	267
33.	Second approximation of operator control in COBWEB/2	268
34.	Norms for congressional classes formed without priming.	283

List of Figures

Figure	Page
1. A model of learning and performance.	2
2. An example classification tree	3
3. Antecedents to COBWEB.	8
4. A view of system development	14
5. An information processing view of methodological perspectives in AI	16
6. Concept induction: identifying a map between objects and concepts	21
7. An example concept space.	26
8. 'Search' for maximally-specific,conjunctive,attribute-value concepts	31
9. The performance task for learning from examples	32
10. An example decision tree	34
11. An example of GBM	37
12. An agglomerative method at work.	42
13. Clustering in RUMMAGE and DISCON.	45
14. Processing in the Partitioning Module of CLUSTER/2.	46
15. Generating clusterings in conceptual clustering.	48
16. Evaluating clustering in conceptual clustering	49
17. Examples for the 'ARCH' program	52
18. Approximating backtracking by 'forgetting' in ID4	55
19. A performance task for conceptual clustering	61
20. Encoding joint probabilities in tree form.	77
21. Concept trees over non-linearly and linearly separable sets.	86
22. Hierarchies of objects defined by outer, inner, and bottom shape. .	96
23. Classification tree over animal descriptions	104
24. Adding 'mammal' and 'bird' to memory	111

25.	The effect of node merging	113
26.	Merging caused by adding 'amphibian-1'.	114
27.	The effect of node splitting	115
28.	Splitting caused by adding 'mammal-2'.	116
29.	Tree transformation by applying <i>split, split, merge, merge</i>	118
30.	A demonstration of three nodes being 'merged' over multiple trials.	121
31.	An example of a lack of heuristic 'foresight'	122
32.	A node and its prototype used to test for node misplacement	123
33.	Outline of the object incorporation process in COBWEB.	130
34.	Schematic of superordinate node merging	131
35.	Basic level node merging with superordinate concepts	132
36.	A classification tree with superordinate concept.	133
37.	Development history and final version of COBWEB.	140
38.	A classification tree over document descriptions	144
39.	Nested classes with defaults.	149
40.	Determining a node where Budget-cuts = 'yes' is normative.	154
41.	Determining a node where Budget-cuts = 'no' is normative	155
42.	Determining subnorms for Budget-cuts.	156
43.	Diagnostic success with soybean cases	163
44.	The top levels of a classification tree over soybean disease cases	164
45.	Success at inference averaged over all soybean attributes.	165
46.	Increase in correct inference as a function of attribute dependence.	167
47.	The top levels of a classification tree over thyroid disease cases.	169
48.	Diagnostic success with thyroid cases.	170
49.	Success at inference averaged over all thyroid attributes	170
50.	Increase in correct inference as a function of attribute dependence.	171
51.	Difference between using and not using first soybean norm	173

52.	Difference between using and not using unconditioned soybean norms	174
53.	Difference between using and not using unconditioned soybean norms	175
54.	Increase in correct inference with norms over frequency approach .	176
55.	Difference between using and not using first thyroid norm	177
56.	Difference between using and not using unconditioned thyroid norms	178
57.	A comparison of ID3 (no chi-square) and COBWEB	180
58.	A comparison of best scores obtained by ID3' and COBWEB. . . .	182
59.	Domains with global and local optimums	195
60.	Convergence on optimal partitions	197
61.	Domains used to test convergence time.	198
62.	Number of objects required to converge	199
63.	Sample nodes with predictability and predictiveness measures . . .	207
64.	A partially indexed tree of animals	210
65.	Attribute-value encoding of the Murphy and Smith tree	215
66.	Indexing imposed on the tree of Murphy and Smith experiment . .	216
67.	Hierarchies of objects defined by outer, inner, and bottom shape. .	217
68.	Indexing imposed on tree 2 of Hoffman and Ziessler Experiments .	220
69.	Partially indexed tree over strings of Table 23	225
70.	Partially indexed (COBWEB) tree over strings of Table 23	227
71.	Human and computer simulated response times	229
72.	Partially indexed (manual) tree over strings of Table 25	232
73.	Partially indexed (COBWEB) tree over strings of Table 25	233
74.	Human and computer simulated response times	236
75.	Recognition time as a function of collocation (typicality).	240
76.	Subject's judgements of typicality and collocation	241
77.	Response times for 'conservative' senators as function of typicality	244
78.	Response times for 'liberal' senators as function of typicality	245

79.	Time to retrieve 'conservatives' in the congressional domain.	247
80.	Difference between basic and subord total predictiveness (thyroid).	251
81.	Difference between basic and subord total predictiveness (liberal)	252
82.	The 'conservative' class with subordinate target concepts	253
83.	Response time as function of typicality to subordinate node (N_2)	254
84.	Response time as function of typicality to subordinate node (N_{36}).	255
85.	Testing the quality of creating a new class in COBWEB/2	265
86.	Merging the two best hosts in COBWEB/2	266
87.	Domains with global and local optimums	274
88.	Convergence on optimal partitions in COBWEB/2	276
89.	An example of indexing 'brittleness'	277
90.	Convergence on optimal partitions using priming in COBWEB/2	278
91.	Domains used to test convergence time.	280
92.	Number of objects required to converge by COBWEB/2	281
93.	Diagnostic success on Soybean Cases without Priming	284
94.	Diagnostic success on thyroid Cases without Priming	285
95.	Increase in correct inference using COBWEB/2 without priming.	286
96.	Average increase in correct prediction afforded by COBWEB	294
97.	Average increase in correct prediction afforded by ID3 Trees	296

Acknowledgements

I thank my dissertation committee for their guidance and friendship during my time apprenticing under them. I will not try to qualify my thanks to Dennis Kibler, Pat Langley, and Rick Granger. Their contribution to this work and my view on many things has been profound. I hope to take their lessons with me. In addition, Tarow Indow fueled much of my early interest in AI and Don Hoffman gave me considerable support when I advanced to candidacy.

I am thankful for the friendship and support of the entire Computer Science Department. The hard work and competence of the computing support group, notably Hieu Tran, Craig Rolandelli, Tim Morgan, Mary Hegardt, and Larry Kaplan, is most appreciated. The administrative staff, particularly Pat Harris, Susan Hyatt, Phyllis Siegal, Mary Day, Sue Rose, Caroline Ehrlich, Judy Hornaday, Lynn Harris, Candy Mamer, Susan Moore, Fran Paz, Marriane Schnaubelt, Essie Lev, Peggy Rose, Laura Shell, and Laura Yoklavich made life a lot of fun. My fellow graduate students, Don Rose, Dale McNulty, Kurt Eiselt, Jeff Schlimmer, Rogers Hall, Chris Pidgeon, Steve Hampson, Paul Morris, Ira Baxter, Guillermo Arango, Julio Leite, Dan Easterlin, Jon Gilbert, Jack Beusmans, Karen Weickert, Jeff Thomas, Amnon Myers, Dave Benjamin, and Tim Shimeall have put up with me the longest and have made life tolerable at worst and ecstatic at best. Kurt, Don, Dale, Rogers, and Jeff(s) have particularly helped me in good times and bad. Other friends, particularly Joan Steicker, but including John Long, Tami Taylor, Craig Snyder, and Marianne Henkes helped me survive the early years of graduate school, while Ken Habaeker and others helped during the latter years – I will always be grateful.

Finally, while sibling rivalry makes it difficult to admit certain things, my office-mate, Jeff Schlimmer, is responsible for much of my success. Pounding on the keys from dawn to dusk six days a week, he finally shamed me into finishing. Besides this, our discussions were very influential and late at night his desk was especially easy pickings. I've tried my best to emulate Jeff, as well as Bruce Porter, Rogers Hall, Etienne Wenger, and Jeff Thomas. Know it or not, along with my dissertation committee, they have been my advisors.

This work has been supported by Contract N00014-84-K-0345 from the Information Sciences Division, Office of Naval Research and a gift from the Hughes Aircraft Company. Some of the work presented in Chapter 2 has been published as "Methods of Conceptual Clustering and their Relation to Numerical Taxonomy" in *Artificial Intelligence and Statistics*, Addison-Wesley, 1985. I thank Addison-Wesley for allowing me to use this material. In addition, I thank Robert Stepp for allowing me to reprint the soybean data from his dissertation, "Conjunctive Conceptual Clustering: A methodology and experimentation". Ross Quinlan graciously made available the previously unpublished thyroid data given in the appendix. Last, I thank Congressional Quarterly Inc. for allowing me to use congressional voting data from "Congressional Roll Call" (pp. 14-15), 1985.

Curriculum Vitae Douglas Hayes Fisher Jr.

- 1957 Born Oakland, California
- 1980 Bachelor of Science in Information and Computer Science,
University of California, Irvine
- 1983 Master of Science in Information and Computer Science,
University of California, Irvine
- 1987 Doctor of Philosophy in Information and Computer Science,
University of California, Irvine

Publications

- "Knowledge Acquisition Via Incremental Conceptual Clustering." *Machine Learning*, 2, 1987.
- "Conceptual Clustering, Learning from Examples, and Inference." *Proceedings of Fourth International Machine Learning Workshop*, Irvine, CA, 1987.
- "Improving Inference Through Conceptual Clustering." *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA, 1987.
- "A Proposed Method of Clustering Structured and Decomposable Objects", *Machine Learning: A Guide to Current Research*, Kluwer, 1986.
- "Methods of Conceptual Clustering and Their Relation to Numerical Taxonomy" (with Pat Langley), *Artificial Intelligence and Statistics*, Addison-Wesley, 1986.
- "A Case Study of Incremental Concept Induction" (with Jeff Schlimmer), *The Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, 1986.
- "Approaches to Conceptual Clustering" (with Pat Langley), *Proceedings of the International Joint Conference on Artificial Intelligence*, Los Angeles, CA, 1985.
- "A Hierarchical Conceptual Clustering Algorithm", Technical Report 85-21, Department of Information and Computer Science, University of California, Irvine, 1985.
- "Discovering Qualitative Empirical Laws" (with Pat Langley, Herbert Simon, and Jan Zytkow), Technical Report 85-18, Department of Information and Computer Science, University of California, Irvine, 1985.

Professional Memberships

American Association for Artificial Intelligence

Professional Service

Proposal reviewer for the National Science Foundation, Data and Knowledge-Base Systems

Reviewer for the journal, *Machine Learning*

Reviewer for the journal, *Artificial Intelligence*

Abstract of the Dissertation

Knowledge Acquisition Via Incremental Conceptual Clustering

by

Douglas Hayes Fisher Jr.

Doctor of Philosophy in Information and Computer Science

University of California, Irvine, 1987

Dr. Dennis Kibler, Chair

Concept learning and organization are much studied in artificial intelligence and cognitive psychology. Computational models of learning and memory that hope to be flexibly applied in real-world settings need to be incremental and improve an agent's ability to make predictions about the environment. While these are useful properties for purely artificial organisms, they also characterize much of human learning and memory.

This dissertation describes COBWEB, an incremental method of *conceptual clustering* that builds a classification hierarchy over a sequence of observations. These hierarchies are characterized in terms of their ability to improve prediction of unknown object properties. Computer experimentation and comparisons with alternate methods of classification show that COBWEB's approach effectively improves prediction ability. More generally, prediction of unknown object properties is forwarded as a performance task for all conceptual clustering systems. This opens the way for objective, not anecdotal, characterizations of and comparisons between concept formation systems.

A fundamental bias of this dissertation is that research on human learning and memory can usefully inspire directions for work on artificially intelligent systems and vice versa. Concept representations and measures of concept quality used by COBWEB are inspired by work in cognitive psychology on *typicality* and *basic level* effects. Conversely, COBWEB is the basis for a second system, COBWEB/2, that accounts for typicality and basic level effects in humans. Apparently, this is the first computational model that accounts for basic level effects. The account of typicality effects stresses the need to consider concepts in the context of a larger memory structure. This approach also facilitates speculation on possible interactions between basic level and typicality effects.

In summary, the dissertation presents an incremental method of conceptual clustering that is evaluated with respect to a prediction task. Concept representations and heuristics are borrowed from cognitive psychology, with repayment in the form of a cognitive model of basic level and typicality effects.

CHAPTER 1

Introduction

1.1 Contributions of the Dissertation

Classification is the basis of inferential capacity and is critical to the success of any intelligent organism. Artificial intelligence (AI) and cognitive psychology seek to explain the form and acquisition of classification structures and processes. This dissertation reports two systems for building classification schemes that have been influenced by principles of AI and cognitive psychology.

From an AI or machine learning standpoint this dissertation addresses the problem of learning under two assumptions. The first is that concept learning is *incremental*; objects are incorporated into a classification structure as they are observed. Second, concept learning should increase the correctness of predictions made about the environment. These assumptions are studied within the context of *conceptual clustering*, a machine learning task concerned with building classification structures.

From a cognitive psychology standpoint, constraints on human classification can illuminate principles of (human and machine) intelligence, generally. In particular, this dissertation takes advantage of research on *basic level* and *typicality* effects observed during human classification. These effects suggest principled ways of measuring concept quality, representing concepts, and classifying objects.

Two foci of interest, psychological and computational, represent apparently dichotomous objectives. However, these interests are cooperative and the interplay between them yields insights that are incorporated into two concept formation systems. COBWEB is an incremental conceptual clustering system that attempts

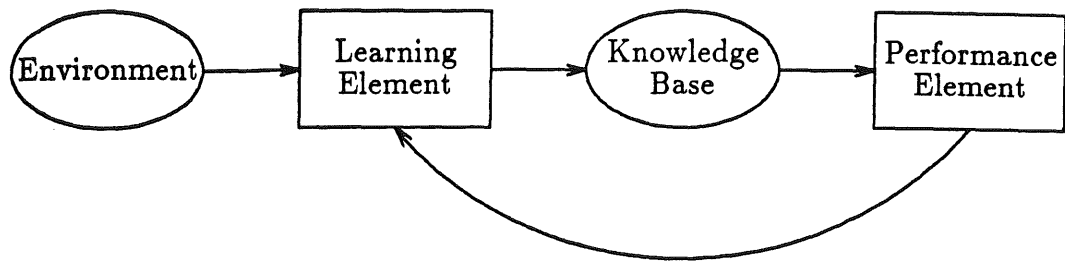


Figure 1

A model of learning and performance

to maximize the ability to correctly predict unknown object properties. To do this it uses a measure of concept quality inspired by psychological studies. Furthermore, incremental processing and inference ability characterize much of human learning and memory. From COBWEB, a second system, COBWEB/2 is derived. This system builds classification hierarchies that account for basic level and typicality phenomena. From a cognitive modeling standpoint, this work appears to be the first computational model to account for basic level effects, and its explanation of typicality effects has several advantages over previous accounts.

1.2 Conceptual Clustering

Machine learning is concerned with improving performance by automating knowledge acquisition and refinement. This view is reflected by the simple model of learning and performance in Figure 1 [DIET82]. Learning organizes observations into a knowledge base that facilitates performance with respect to some task. Assumptions about environment, knowledge base, and performance all impact the design of a learning algorithm and delineate general learning tasks. For instance, *learning from examples* assumes that objects (states, events, etc.) come preclassified with respect to a number of 'teacher' defined classes. Under this environmental assumption a learner induces concepts for each object class. Learning to diagnose soybean disease from examples [MICH81] assumes that a 'teacher' identifies the

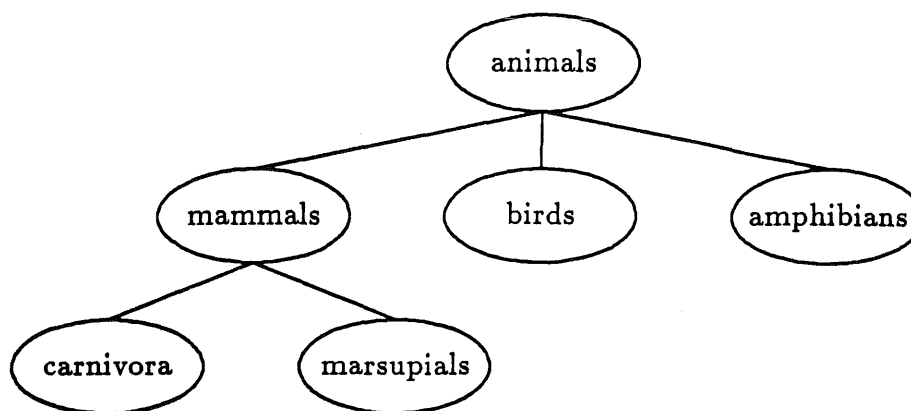


Figure 2

An example classification tree

disease (or lack of disease) of soybean plant case histories. Over several case histories the learner induces rules or concepts that allow it to independently identify diseases categories. Learning from examples has been applied in numerous domains [WINS75, HAYE78, VERE80, MITC83, PORT84, BRAD87, SCH86B], but in every system that learns from examples, performance reduces to matching previously unseen 'objects' against induced concepts, thus identifying their class membership (e.g., an example of a particular soybean disease).

In contrast to learning from examples, *conceptual clustering* systems [MICH80] accept a number of object descriptions and produce a classification scheme over the observed objects. For example, a conceptual clustering system might form a classification tree over a number of animal descriptions as shown in Figure 2. These systems do not require a 'teacher' to preclassify objects, but use an evaluation function to find classes with 'good' concept descriptions. Concept descriptions may be stored at classification tree nodes. For example, the 'mammals' node of Figure 2 might be characterized by the concept, $\text{has-hair} \wedge \text{bears-living-young}$. Conceptual clustering is a type of *learning by observation* or *concept formation* (as

opposed to *learning from examples*). However, the recency of conceptual clustering's definition has allowed little exploration of it in the context of environment and performance.

The most important contextual factor surrounding learning is the performance task that benefits from it. Unfortunately, this task is ill-defined or not discussed at all with respect to most conceptual clustering work (and thus the often asked question, "How do you know the classifications you get are any good?"). However, some attempts have been made to evaluate conceptual clustering with respect to a performance task. For example, Cheng and Fu [CHEN85] and Fu and Buchanan [FU85] use clustering techniques to facilitate disease diagnosis in expert systems. Generalizing (and clarifying) their use of conceptual clustering, classifications can be the basis for effective prediction of unseen object properties. The generality of classification as a means of guiding inference is manifest in recent discussions of problem-solving as classification [CLAN84]. For example, having recognized an animal with respect to the 'mammals' node of Figure 2 – say by virtue of it having hair – a prediction that it bears-living-young can be made. In a medical domain, a set of symptoms may suggest a particular disease, from which a treatment can be inferred. The first system described in this dissertation, COBWEB, is designed to form classification trees that are good predictive models of the environment.

A second factor surrounding learning is the environment. In particular, conceptual clustering systems have assumed that environmental inputs are indefinitely available for examination and thus the environment is amenable to nonincremental processing of observations. However, real world environments encourage incremental object assimilation [CARB86, LAN86A, SAMM86] and systems that process observations in this fashion are gaining prominence [REIN85, SCH86A, LEB082, KOL83A]. In response to real world considerations, COBWEB has been constructed

as an incremental system of conceptual clustering. Its underlying control mechanisms are abstracted from previous work on incremental concept formation, notably Lebowitz' UNIMEM [LEBO82] and Kolodner's CYRUS [KOL83A]. However, unlike these precursors, COBWEB is evaluated along a number of dimensions related to the cost and quality of learning.

This dissertation imposes the framework of conceptual clustering onto incremental concept formation systems like those developed by Lebowitz [LEBO82] and Kolodner [KOL83A]. This combination extends the traditional conceptual clustering literature to include incremental processing and clarifies the processing characteristics of these other incremental systems. In addition, this work suggests prediction of missing object properties as a performance task for conceptual clustering.

1.3 Basic Level and Typicality Effects

The processing strategies of COBWEB borrow from work in AI and machine learning. However, the AI influence is balanced with results from cognitive psychology. Many aspects of human intelligence demonstrate important principles of general intelligence. In the context of classification, two phenomena are of particular interest. The first is that members of a class are not regarded as equally representative, but vary along a dimension of *typicality* [MERV81, SMIT81]. For example, a *robin* is more quickly recognized as a *bird* than is a *penguin*. The observation that some instances are regarded as more typical of a class than others does not jibe with assumptions often associated with logical, typically conjunctive, concepts [SMIT81]. The limitations of logical representations motivates the use of *probabilistic concepts* in COBWEB. Probabilistic concepts associate probabilities or other confidence measures with object class properties. For example, a platypus is a mammal that lays eggs. A probabilistic concept for 'mammals' would indicate

that a mammal has-hair with probability, 1.0, and bears-living-young with probability, 0.97, to accommodate platypi. Generally, probabilities add information to concept representations that can be exploited during classification and inference.

Other studies indicate the tendency of humans to prefer a particular conceptual level in hierarchical classification schemes [ROS76A, JOLI84]. For instance, when asked to identify a *collie*, a subject will respond that it is a *dog* rather than a *collie* or *mammal*. This task, and a host of others [MERV81], indicate that in a hierarchy containing (*collie*, *dog*, *mammal*, *animal*), *dog* is the preferred or *basic level* concept. The identification of preferred concepts in humans suggests principled measures of concept quality in AI systems. COBWEB uses a measure of concept quality called *category utility* [GLUC85] that was inspired by basic level studies. Category utility assumes probabilistic information is known regarding class members, thus reinforcing the choice of probabilistic concepts. Moreover, category utility rewards concepts that facilitate prediction and is therefore compatible with COBWEB's performance goals.

Basic level and typicality effects motivate concept representation and evaluation in COBWEB. These psychological considerations do not interfere with the computational goals of incremental processing and utility of classifications for inference. Rather, probabilistic concepts and category utility are completely compatible with these goals.

Although its design is influenced by psychological concerns, COBWEB should not be regarded as a cognitive model *per se*. However, its environmental (i.e., incremental processing), performance (i.e., inference), and knowledge base (i.e., hierarchical classifications and probabilistic concepts) assumptions are consistent with much of human learning and memory. As a result, the memory structures of COBWEB are the basis of a second memory model that accounts for typicality and basic level effects observed during human classification. These hierarchies use an

indexing scheme adapted from work by Lebowitz [LEBO82] and Kolodner [KOL83A] and they demonstrate how various pieces of partial evidence combine to produce the desired psychological effects. Indexed classification hierarchies are learned by COBWEB/2, a second system that demonstrates advantages and problems with indexed memory when learning.

1.4 An Overview of Computational and Psychological Antecedents

In summary, this dissertation draws upon work from AI and cognitive psychology. Work in conceptual clustering and incremental concept formation contributes to COBWEB's and COBWEB/2's control mechanisms, while work in cognitive psychology suggests concept representations and quality preferences. Specific antecedents and their contributions to this work are pictured in Figure 3. While interest in natural versus artificial intelligence traditionally divides research efforts in AI [HALL85, NEWE73], they are intertwined in this dissertation.

1.5 Methodological Biases

AI is an evolving discipline, amalgamating concepts from several fields, including computer science and psychology. As a result there is no consensus among AI practitioners as to which research problems are important, which methodologies are productive, and in general, what constitutes 'good research' [HALL85]. The burden of identifying important questions, productive methodologies, and evaluation criteria is placed on the individual researcher; these are not explicit and well understood constraints of the field as a whole. However, research communities have emerged within the field, the identification of which can aid in guiding and reporting research.

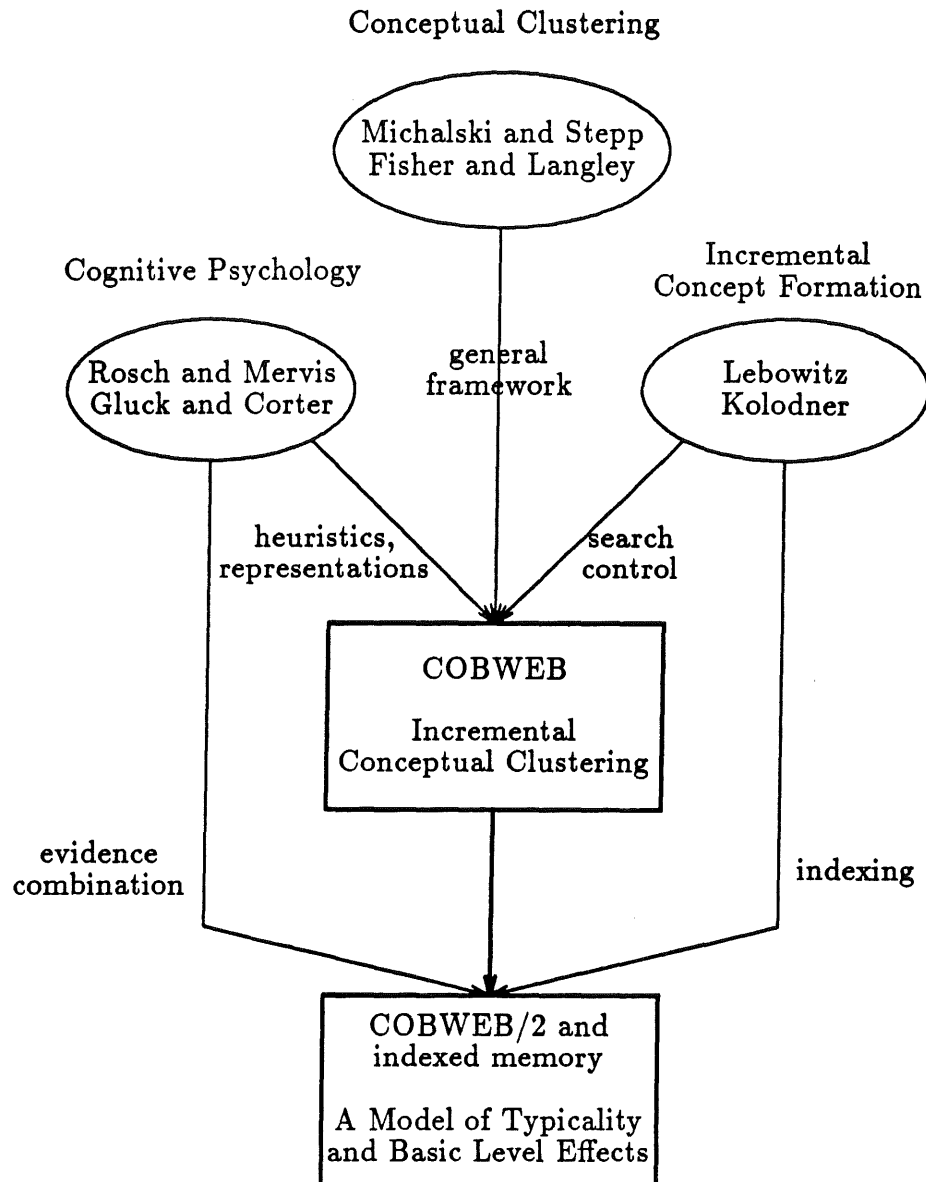


Figure 3

Antecedents to COBWEB

1.5.1 Methodological Perspectives in AI

Hall and Kibler [HALL85] have recently forwarded a taxonomy of methodological perspectives in AI. They initially divide perspectives by interest in natural intelligence versus purely artificial intelligence. Natural approaches are further

broken down by the specificity of the natural behavior that a computer system is expected to model.

Empirical approaches are characterized by a validation of system behavior with respect to tightly constrained experimental evidence. Experiments need not be performed by the 'cognitive modeler', but may be performed previous to and independent of the computational model. An existing database of experimental findings can be used for comparative purposes. Hall and Kibler cite GPS [NEW63] as an example of this perspective. Feigenbaum's EPAM [FEIG63] is another.

In contrast, *speculative* approaches look to natural behavior for initial inspiration, introspect as to the rules guiding this behavior, and validate the resultant computer system by gross comparison of system and natural behavior. Speculative methods are not constrained by specific experimental evidence, but seek general principles by looking to 'general' behavior. Hall and Kibler give Schank and Abelson's [SCHA77] theory of scripts as an exemplar of the speculative approach.

Empirical and speculative approaches can be viewed as differing in the 'grain size' of the natural phenomena that are used to validate the cognitive model. The empirical approach dictates validation with respect to tightly constrained behavior, while research efforts following the speculative approach are compared with natural behavior of less specificity. Importantly, this distinction does not imply that the *mechanisms* of an empirical artifact be special purpose. In fact, one property (intended or not) of many empirically motivated studies is that the cognitive model's mechanisms move beyond the experimental evidence and allow predictions about natural behavior that was not the original focus of study. More generally, mechanisms suggested by either perspective may be transported outside the realm of psychological interest entirely. The means-ends strategy of GPS and semantic nets [QUIL68] are well-known examples of formalisms that

were initiated for cognitive modeling, but that have been adopted by artificial intelligence generally.

In contrast to studies of natural behavior, Hall and Kibler propose three perspectives interested in strictly 'artificial' intelligence. *Constructive* AI forces general principles of intelligence to emerge by designing and building computer systems that address complex but specific real-world problems. For example, DENDRAL [BUCH69] illuminated general issues of knowledge-intensive or expert systems while focusing on the specific task of identifying molecular structure.

Analysis of heuristic search (A^*) [HART68] is an example of work in the *formal* perspective of AI. In general, formal work seeks to unify a body of disparate work under a single, generalizing framework. Additionally, Hall and Kibler stipulate that this unifying framework be characterized formally or analytically (e.g., by proofs of correctness).

Finally, *performance* AI seeks to achieve expert behavior, with little concern towards extracting important processing principles that underlie performance. Performance AI should not be identified with every system concerned with a performance task, but only with systems that are concerned with performance to the exclusion of underlying processing principles.

1.5.2 The Dissertation in Perspective

This dissertation reflects several of the approaches outlined by Hall and Kibler. With important qualifications, the development of COBWEB resembles a formal study. The conceptual clustering framework proposed by Michalski and Stepp [MIC83A, MIC83B], and elaborated by Fisher and Langley [FIS85A, FIS86A], clarifies the basic control mechanisms of existing incremental concept formation systems [LEBO82, KOL83A]. This inspires the basic processing assumptions of COBWEB. In addition, the system uses probabilistic concepts and a principled

measure of concept quality, as opposed to logical representations or the statistically-based, but *ad hoc*, representations and measures found in [LEBO82, KOL83A]. This approach highlights issues of representation and evaluation that are difficult to extract from more *ad hoc* techniques and also suggests several dimensions for evaluating similar systems. Prediction of unseen object properties is implied as a performance task for conceptual clustering systems and criteria relating to the cost and quality of learning are suggested as dimensions for evaluating incremental concept formation systems.¹

Like formal approaches, work on COBWEB seeks to clarify and cast new light on existing work. However, the characterization of COBWEB is not analytical, but relies instead on empirical validation via extensive computer experimentation. This type of process characterization is influenced by Quinlan [QUIN86] and others [HAMP83, SCH86A] and finds its roots in work on pattern recognition and data analysis [DUDA73, EVER80]. However, COBWEB's empirical characterization is novel in several respects, most notably as it relates to prediction ability. The system's ability to make accurate predictions is compared to two alternative methods: a 'straw man' and a better known system for learning from examples.

Furthermore, COBWEB is not only characterized in a number of domains, but a measure for characterizing the domain itself is forwarded. In general, little attention has been paid to Simon's point [SIMO69] that domains must be characterized before the advantage of a learning system can be evaluated. Collectively, computer experiments are used to address the same issues as more formal methods, e.g., system behavior under varying conditions. There is no debate that when possible, a formal analysis is better than an empirical one. But when a system (or

¹ Importantly, control strategies and representations used by COBWEB were abstracted from or inspired by existing systems; they did not emerge (in this study) as the result of exploring concept formation in a highly constrained domain. Thus, COBWEB's development is not constructive. However, the development of some of COBWEB's precursors, particularly UNIMEM and CYRUS, might be so classified.

researcher) is not amenable to formal analysis, empirical studies may enumerate the dimensions along which some future, more formal analysis can proceed.

Although COBWEB is of computational interest, the work reported in the following chapters also reflects empirical and speculative approaches to cognitive modeling. In particular, COBWEB classification trees are the basis for an indexed memory structure that accounts for certain basic level and typicality effects. While this dissertation does not include psychological experimentation, a significant body of existing research supports the existence of these phenomena. However, there are difficulties with using all of this data for comparative purposes. Many experimental studies use natural domains (e.g., animals), but in such domains there is no way of knowing the properties that human subjects use to represent instances and therefore no way of assuring equivalent encodings in the computer model. Nonetheless, comparisons between human subjects and the computer model are made with respect to two experimental studies of basic level effects and one study of typicality effects, each using artificially constructed domains (e.g., nonsense strings). Artificial domains allow some experimental control over the properties to which subjects attend.

Besides the three experimental studies referred to above, the cognitive model is also characterized with respect to other tasks and domains, but these comparisons are hypothetical in nature. For example, computer experiments using a classification hierarchy over objects of the 'natural' domain of congressional voting records suggest several properties of human memory that cannot currently be verified as consistent with human behavior.

Additionally, the cognitive model is characterized with respect to some of the same tasks as COBWEB classification trees. Incremental learning and accurate prediction are important to human behavior and the ability to do these well is the classic sort of evidence admitted by speculative studies for the legitimacy of

a cognitive model. However, this dissertation avoids using such an analysis as confirming evidence for the psychological validity of a model. Rather, the bias is that these analyses supply disconfirming evidence, if they supply any evidence at all. If indexed memory can not be effectively modified as learning occurs this must impact claims for its psychological validity, as well as its computational utility. In particular, the learnability of indexed classification hierarchies is investigated in the context of COBWEB/2, a system derived from COBWEB. Analysis of COBWEB/2's behavior generally indicates good learning and prediction abilities, but also uncovers a weakness of indexing. The impact of this finding on the validity of the memory model is discussed.

The methodological biases exhibited in this dissertation have been related to three of the approaches outlined by Hall and Kibler: formal, empirical, and speculative. However, problems arise when one uses their taxonomy to classify the biases of the dissertation. For example, while work on COBWEB reflects the intent of the formal approach, empirical, rather than formal characterization distinguishes it from this approach. Furthermore, the dissertation addresses computational as well as psychological concerns. This dichotomy is magnified by Hall and Kibler's initial division of methods by the intention of the researcher (i.e., interest in natural versus artificial intelligence). This division is common to other commentaries on methodological biases in AI (e.g., [NEWE73]) as well. A taxonomy that generalizes the formal approach and lessens the apparent schism between computational and psychological research is motivated and developed next.

1.5.3 A Taxonomy of AI Research

Classification schemes are rarely useful if developed in a vacuum; typically, they are motivated by some intention or goal. A fundamental bias of this dissertation is that AI depends on demonstrations of natural intelligence to supply specifications for its systems, whether this is inspirational or is more constrained

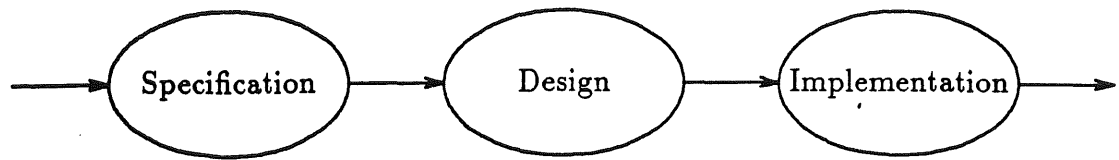


Figure 4

A view of system development

in nature. Therefore, division of research based on artificial versus natural intelligence is somewhat illusory. It may also be counterproductive since it may mask insights into methods and representations that are true across natural and artificial systems (e.g., the utility of probabilistic concepts). An alternate to the intention-centered taxonomy of Hall and Kibler is a method- or design- centered taxonomy that emphasizes the portability or generality of an *information-processing system*.

A popular information-processing view of system development distinguishes *specification*, *design*, and *implementation* (Figure 4) [PAGE80]. The specification is a statement of a system's function. Whether the system is a cash register, a library access system, or an expert system for medical diagnosis, system specification describes *what* the system is supposed to do; specification defines a 'black box'. The objective of system design is to outline the procedures and data representations necessary to satisfy the functionality of a black box. That is, design is concerned with *how* a system performs. Finally, implementation is concerned with realizing procedures and representations on a physical device (e.g., a computer). Within AI proper, Marr's *computational theory*, *algorithm*, and *implementation* levels [MARR82] represent a similar view of information processing systems.² Analogs to Hall and Kibler's perspectives can be understood in terms of how they differ along dimensions suggested by this view of information processing systems.

² Of course there are problems with an exact mapping between these views. In particular, Marr is intimately concerned with 'why' a computation occurs. This issue typically arises in a *requirements analysis* phase of system development, which precedes specification.

Four perspectives seek to uncover general principles or descriptions of intelligence that best fit the design level of information-processing systems. Heuristic search, evidence combination, scripts, and means-ends analysis are general processing principles that emerged from work in the formal, constructive, speculative, and empirical perspectives, respectively. Only performance AI is unconcerned with forwarding general processing principles.

Constructive, empirical, and performance AI each assume domain-dependent specifications. This assertion rests on the assumption that an 'expert' identifying molecular structures and a subject solving the eight-tile puzzle exhibit behavior of roughly the same granularity or level of specificity, despite differences in the overall complexity of these tasks. GPS and much of the work on expert systems move from specific (specifications) to general (designs).

Formal and speculative approaches are distinguished by their use of general, domain-independent specifications. Note that this does not imply ill-defined specifications. For example, A* is precisely specified. Furthermore, objections to (apparently) speculative approaches [OHLS83] may be symptomatic of 'bad speculation' and not of the speculative approach itself.

Figure 5 gives a revised taxonomy of AI perspectives. This taxonomy is heavily influenced by, but differs in a number of respects from Hall and Kibler's framework. At the top level, perspectives are divided in terms of the generality of system design or principles. Performance AI is distinguished from the others along this dimension. The remaining four perspectives are distinguished by the generality of specification or problem statement. Constructive and empirical approaches move from specific problem statements to general principles, while formal and speculative approaches assume that general mechanisms/representations are derived from general, domain-independent specifications. Finally, approaches are distinguished by the interest of the researcher in natural versus purely computational intelligence.

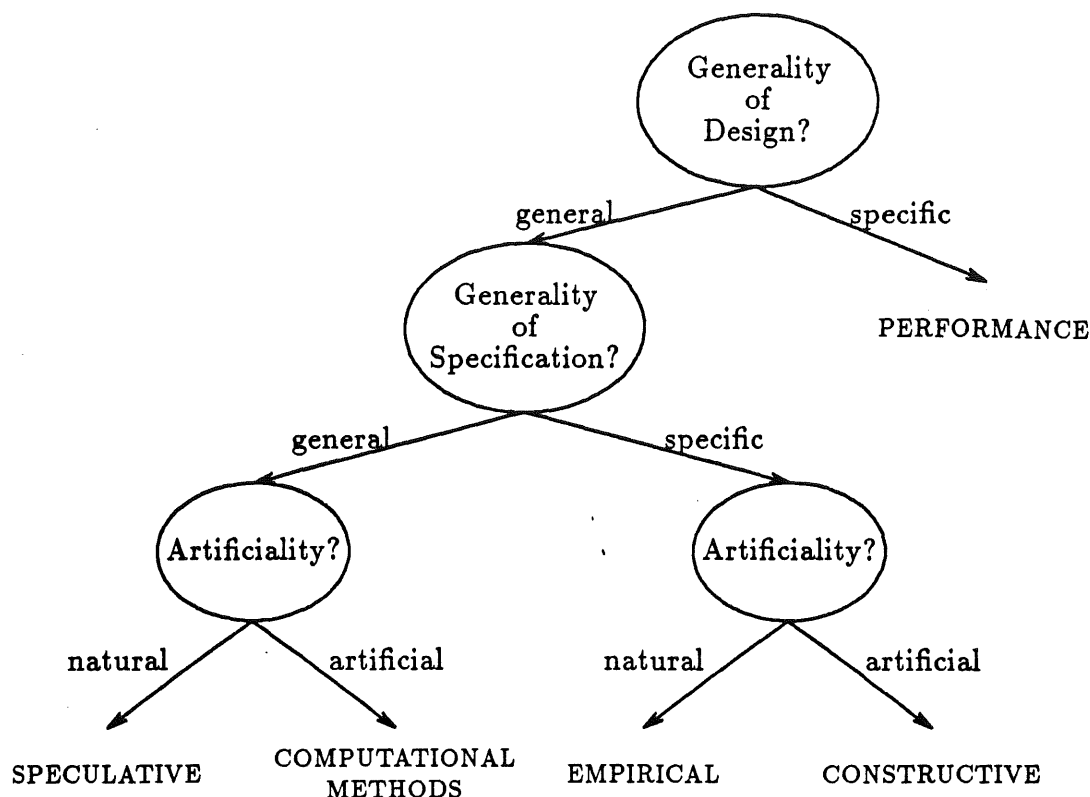


Figure 5

An information processing view of methodological perspectives in AI

At the leaves of this taxonomy, the formal approach has been generalized to one that roughly corresponds to Newell's view of AI as a quest for general 'methods' [NEWE73]. This approach shares the intent of Hall and Kibler's formal approach, but leaves the form of characterization (i.e., formal or empirical) unspecified.

The taxonomy of Figure 5 demotes the importance of the natural/artificial distinction. Rather it emphasizes the importance of design and specification generality, and thus the portability of ideas across domains and tasks. Importantly, few researchers fit precisely within one perspective. However, this taxonomy predicts that a researcher's differing perspectives will share method; constructive and empirical approaches will intermingle, as will the computational methods and speculative

approaches. This seems to more accurately reflect the sort of methodological shifting that occurs than does a taxonomy that initially differentiates based on artificial and natural orientations. Hall and Kibler cite Feigenbaum's early work on EPAM [FEIG63] as an example of work in the empirical perspective, while his later work on expert systems is constructive. Closer to home, this taxonomy lessens the schism between interest in natural and computational mechanisms that is exhibited in this dissertation. Hall (personal communication) suggests that combining interests in natural and artificial intelligence can be problematic, since it facilitates confusions between claims of psychological validity with claims for computational utility. That this happens frequently can be taken as evidence for the descriptive accuracy of the taxonomy of Figure 5. Prescriptively speaking though, this sort of confusion is a flaw that the dissertation seeks to avoid.

1.6 Overview of the Dissertation

This dissertation describes COBWEB, COBWEB/2, and the classification structures formed by these systems. The presentation focuses on the design (or algorithmic) level, as opposed to implementation level descriptions. The emphasis on design-level issues clarifies the connection between these systems. This descriptive level also maximizes the 'portability' of these systems and facilitates *rational reconstruction* [BUND84].

Chapter 2 gives relevant background from machine concept learning. While this chapter describes particular systems, the goal is to present a general framework for incremental conceptual clustering. This framework is described in terms of a predominant AI paradigm: *search*. In addition, the chapter motivates and describes a performance task for conceptual clustering.

Chapter 3 describes important background from cognitive psychology on typicality and basic level effects. Results presented in this chapter are important

for validating the psychological consistency of an indexing scheme presented in chapter 7 and for motivating the concept representation and evaluation measure used during conceptual clustering.

Chapter 4 describes COBWEB, an incremental and domain independent system of conceptual clustering. This system instantiates the general search framework of chapter 2.

Chapter 5 evaluates classification schemes produced by COBWEB in terms of prediction. In particular, experiments with soybean and thyroid disease diagnosis demonstrate the cost effectiveness of the approach as opposed to selected alternative methods.

Chapter 6 characterizes COBWEB along dimensions that are relevant to incremental learning systems. This chapter demonstrates that the system is computationally economical, while still robust in the sense that 'high quality' classification schemes are typically constructed.

Chapter 7 shows how COBWEB classification schemes can be modified to account for basic level and typicality effects. While results from three (human) experimental studies are explained by the classification model, support of a more hypothetical nature is garnered from (computer) experiments in the domain of congressional voting records.

Chapter 8 describes COBWEB/2, a derivative of COBWEB that incrementally builds the classification structures of chapter 7. The system's economy, robustness, and inference ability are characterized in relation to the 'ideal' COBWEB system. The fact that classification structures of this type can be learned and perform reasonably along a number of computationally important dimensions is not taken as confirming evidence for the psychological validity of the indexing scheme. Rather, problems during learning motivate a discussion of some possible weaknesses of the indexing scheme as a psychological model.

Chapter 9 concludes the dissertation with a summary of results and a prospectus of future research.

Chapter Acknowledgements

Organizing frameworks are very general, but nonetheless useful for thought and communication. Dieterich's context for learning is one such framework. So is Hall and Kibler's discussion of methodological perspectives in AI. Both frameworks were important in organizing the chapter and the dissertation as a whole. Discussions with Rogers Hall, Dennis Kibler, Pat Langley, and Jack Beusmans clarified my understanding of existing methodological perspectives in print, but they should not be held responsible for my views on these matters. Discussions with Dennis Kibler are responsible for linking conceptual clustering and performance.

CHAPTER 2

Concept Induction in Artificial Intelligence

2.1 Chapter Overview

This chapter gives background from machine learning that contributes to the development of COBWEB and COBWEB/2. In particular, conceptual clustering is distinguished from other types of concept learning in terms of environmental (input), knowledge base (output), and performance assumptions. Concept learning methods are also differentiated by the learning mechanisms that support differing assumptions. The chapter frames these mechanisms in terms of a pervasive paradigm of AI - *search*.

Mitchell [MITC82] has proposed the view of concept learning as search in the context of *learning from examples*. This task is characterized as a search for concepts that appropriately describe 'teacher' defined classes. Section 2 briefly surveys object and concept languages that can be used to delimit the search space traversed during learning from examples, but focuses on a simple language of attribute-value descriptions; this language is an antecedent to those used by COBWEB and COBWEB/2.

Section 3 describes the task of learning from examples in terms of search. Performance tasks associated with learning from examples reduce to identifying objects with respect to 'teacher' defined classes.

Section 4 extends the search framework to cover methods of conceptual clustering. These methods drop the assumption of a teacher. Many systems also build hierarchical classification schemes, as opposed to the flat concept organizations of learning from examples.

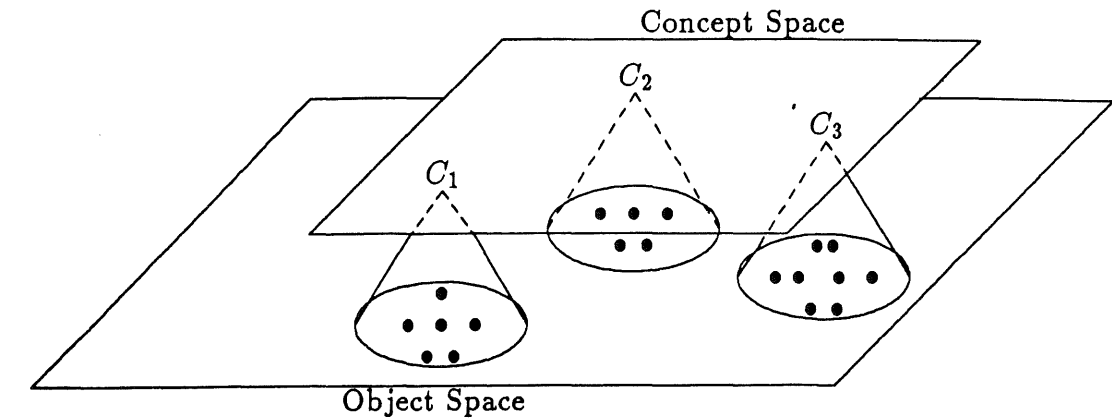


Figure 6

Concept induction: identifying a map between objects and concepts

Many concept induction tasks of the learning from examples and conceptual clustering variety are *nonincremental*. They assume that all objects over which induction occurs are present from the outset of system execution. Section 5 considers incremental induction in terms of search. A class of incremental methods are identified, of which COBWEB and COBWEB/2 are members. Systems of this class share a *hill-climbing* search strategy. Dimensions for evaluating these methods and incremental systems generally are presented.

Finally, section 6 gives a performance task for conceptual clustering that will be used to evaluate COBWEB and COBWEB/2 in later chapters. This task consists of predicting unknown object properties and generalizes the performance task associated with learning from examples – i.e., predicting the ‘special’ property of teacher-defined class membership.

2.2 Object and Concept Representation

Figure 6 illustrates that the task common to all forms of concept induction concerns creating of a map between two languages: an *object* (or *instance*) language

Attributes	Domains
Color	{blue, red, green }
Size	{large, medium, small }
Shape	{sphere, block, wedge }

Table 1

Sample attributes and domains

and a *concept* language. The external world is perceived in terms of an object language, where each string represents a single primitive object (observation, event, etc.). Rather than storing all objects in some extensional fashion, it is desirable to summarize observations in terms of a concept language. In general, a string in the concept language (henceforth, simply *concept*) is a generalized description of some number of objects, and thus intensionally represents the set.

2.2.1 Syntax and Semantics

Many object languages have been used for concept learning. They differ along a continuum of complexity where two (fuzzy) endpoints are the languages of *attribute-value* descriptions and *structural* (or *relational*) descriptions. An attribute is a typed variable that takes on one of several values. These values are termed the *domain* of the attribute. Some example attributes and their respective domains are given in Table 1.

Attributes are similar to unary predicates. For example, Color(red) is true or false of an object. However, throughout the dissertation at most one attribute value will be allowed for a given object. Thus, Color(red) precludes the possibility of Color(blue). In this way, the dissertation's assumptions about attribute-value representations are more restrictive than unary predicates. Allowing an object

-
- 1) {[Color=blue], [Size=large], [Shape=sphere]}
 - 2) {[Color=blue], [Size=medium], [Shape=sphere]}
 - 3) {[Color=blue], [Size=small], [Shape=block]}

Table 2

Objects represented as sets of attribute-value pairs

to have multiple values along the same attribute might be useful in many domains. For example, it would be nice to include Keyword(conceptual-clustering) and Keyword(inference) as well as Author(Fisher) in an 'object' representation of this dissertation. However, this luxury is not assumed in future discussion.

Attribute - value representations admit only unary predicates. However, structured descriptions generally allow arbitrary predication of values. For example, an off-road vehicle might be partially described by the predicates: Bigger-than(wheel-1, wheel-2), Bigger-than(wheel-3, wheel-4), On(axle-1, wheel-1, wheel-3), On(axle-2, wheel-2, wheel-4). 'Values' (e.g., wheel-1) are commonly interpreted more liberally as object *components*.

The choice of an object language has important implications for learning. Structured representations have greater descriptive power, but are more costly to process. In particular, determining the truth of a set of attribute values is a simple matter of independently determining the truth of each. However, the truth of a structured description may depend on interactions between predicate arguments. This introduces nondeterminism that underlies the *matching problem* [MITC82, WINS75] for structured descriptions. A more general overview of issues relating to structured descriptions is given by Nilsson [NILS80].

While structured descriptions are used in a number of systems that are discussed later, attribute-value representations are particularly relevant to the development of COBWEB. Objects will be represented as a set of attribute-value

pairs (or *feature vector*) throughout the dissertation. Table 2 gives three object descriptions. Given attribute value representations of individual objects, some representation of object classes is required. A simple enumeration of objects is an extensional representation of a class. However, the attribute-value language can be extended to allow intensional representations or concepts. A standard extension is to define a concept as a set of attribute-value *set* pairs. For example, consider the following concept:

$$\{[\text{Color} = \{\text{blue, red}\}], [\text{Size} = \{\text{large}\}], [\text{Shape} = \{\text{sphere, block}\}]\}$$

The semantics of this concept are:

$$\{x | x \text{ is an object for which}$$

$$\begin{array}{l} \text{Color} = \text{blue} \vee \text{red} \\ \wedge \text{ Size} = \text{large} \\ \wedge \text{ Shape} = \text{sphere} \vee \text{block} \end{array} \}$$

Thus, there is a simple map between a concept and the set of objects that it *covers*. Allowing value sets permits *internal disjunction* [MICH80], since it represents a disjunction of values within (internal to) an attribute. If a value set is a singleton, its solitary value can be regarded as necessary for concept membership.

To simplify discussion, an object description will be regarded as a concept where all value sets are singletons. This implies that all object descriptions are concept descriptions, but not vice versa. Most systems assume that the object language is a subset of the concept language. This strategy is sometimes called the 'single representation trick' [DIET82].

A short-hand representation sometimes omits an attribute reference in a concept. If an attribute is not explicitly given in a concept then it means that a member of the concept can possess *any* value in the domain of the omitted attribute. This is a process of *dropping conditions* that are not criterial in delimiting concept

-
- 1) {[Color={blue}], [Size={ large,medium,small}], [Shape={sphere,block}]}
- or
- 2) {[Color={blue}], [Size={ large,medium,small}], [Shape={sphere,block,wedge}]}
- or
- 3) {[Color={blue,red,green}], [Size={ large,medium,small}],
[Shape={sphere,block,wedge}]}

Table 3

Concepts of increasing generality

membership. This definition of a concept is similar to the definition of a *conjunctive concept* found in Bruner, Goodnow, and Austin [BRUN56].

2.2.2 A Partial Ordering On Concepts

There may be many concepts that cover a given object set. Consider the following objects:

$$\{[Color=\{blue\}], [Size=\{large\}], [Shape=\{sphere\}]\}$$

$$\{[Color=\{blue\}], [Size=\{medium\}], [Shape=\{sphere\}]\} \quad (2-1)$$

$$\{[Color=\{blue\}], [Size=\{small\}], [Shape=\{block\}]\}$$

The concepts of Table 3 each cover this set. Dropping conditions gives syntactic variants of these concepts:

1) {[Color = {blue}], [Shape = {sphere, block}]}

2) {[Color = {blue}]}

3) {}

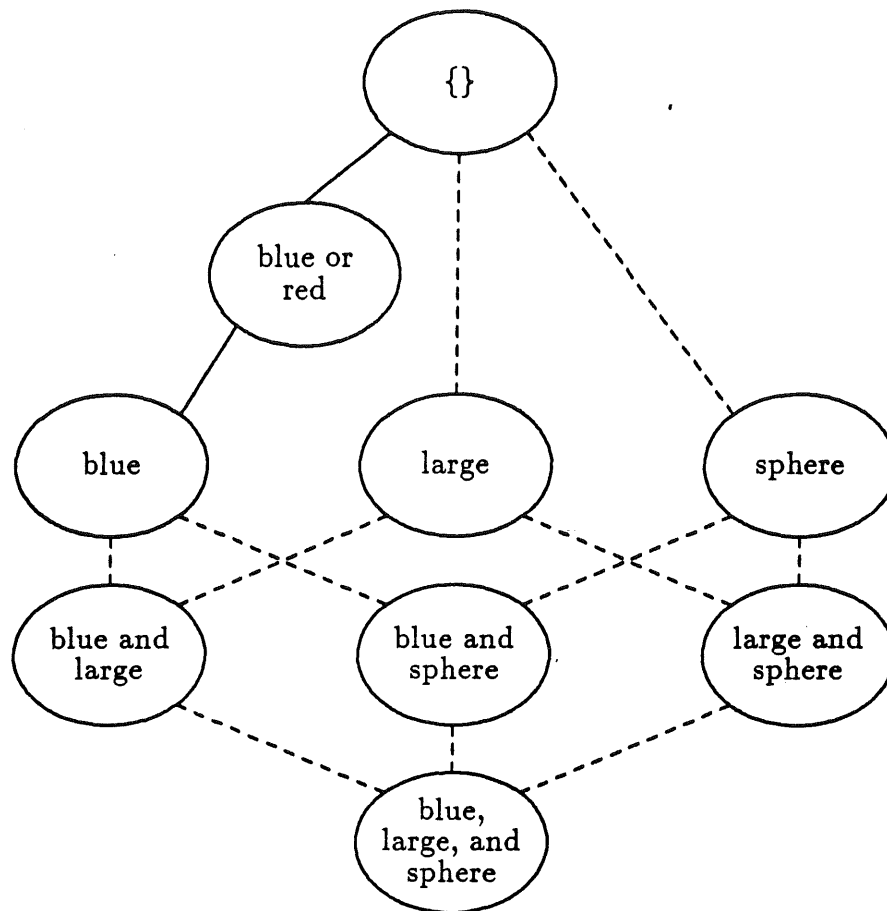


Figure 7

An example concept space

Although each concept covers the object set, concept 3 covers a greater number of possible objects than concept 2, which in turn has greater coverage than concept 1. Letting O_1 , O_2 , and O_3 be the object sets covered by concepts 1, 2, and 3, respectively, then

$$O_1 \subset O_2 \subset O_3.$$

In other words, concepts are *partially ordered* by the subset or *more general than* relation [MITC82]. At the top end of the generality scale is a single *maximally-general* concept that is the empty set (after dropping all conditions). The empty set

covers all possible objects, while at the bottom end are concepts of least generality or *maximal specificity*, which represent single objects. Figure 7 shows a partially enumerated space of concepts defined over Color, Shape, and Size. A shorthand for concepts is used in this figure that expresses $[\text{Shape} = \{\text{blue, red}\}]$ as *blue or red*. In addition, dashed lines indicate that all nodes of the concept space are not shown. For example, *(blue or red) and sphere* is more general than *blue and sphere* but less general than *sphere*, and thus lies between them in the concept space.

The *more general than* relation imposes order on the space of concepts. Concept learning systems can take advantage of this fact when looking for concept descriptions that cover observed objects.

2.3 Learning From Examples

Learning from examples (also *concept identification* or *acquisition*) identifies concepts for teacher-defined object classes. Traditionally, learning from examples has been the prevalent form of machine learning studied. The input/output assumptions of learning from examples are:

- Given:
- A set of objects, O
 - A partition of O , $P = \{O_1, O_2, \dots, O_n\}$, such that $\cup_i O_i = O$
 - A set of concept descriptions, C , which is usually represented intensionally by a concept language.
- Find: A Concept, C_i in C , for each object set, O_i , where each C_i *completely* covers O_i and *consistently* excludes objects in all other O_j . In this case, C_i is said to be a complete and consistent concept [Mic83c].

Each member of a class is a *positive* instance of that class and a *negative* instance of all other classes. Learning from examples assumes that the P_i 's are mutually-disjoint and that corresponding C_i 's cover mutually-disjoint classes as well.

The task of learning from examples is defined by the environmental assumption that objects come preclassified by a teacher. In addition, a property of most

systems is that the knowledge base is a flat set of concept descriptions, although there is a notable exception [QUIN83].

2.3.1 The Learning Element

Under these input/output assumptions, the process of learning from examples has been characterized as a process of search by Mitchell [MITC82]. In this framework, a search space is maintained for each object class. An obvious strategy is to traverse the space of all concepts and retain those that cover the object class and do not intersect any other class. The problem with this strategy is that the concept space may be infinite or prohibitively large. Rather than maintaining an explicit space of concepts, only certain portions of the space are enumerated as they are deemed relevant. More generally, many problems can be solved by first formulating a space of possible *states*, S , and enumerating portions of the space until a solution to the problem is found. This process is termed *state-space search* and it is ubiquitous in AI. Three pieces of information that enable a state-space search to proceed include:

- A selected state from the space of possible states, S , termed the *initial state*.
- A selected subset of S , termed the *goal states*.
- A set of *operators* that transform one state into another. Each operator may have preconditions that must be satisfied by a state before the operator can be applied.

Initial states plus operators give an implicit representation of the entire set of possible states. Beginning with an initial state, operators can be used to transform or expand this state, as well as subsequently generated states. Repeated state expansion traces out a search. Search terminates when a goal state is generated (successful termination), or when there is no state to which any operator applies (unsuccessful termination). A state to which no operators apply is called a *dead-end*.

In searching a state space, operators are usually not applied arbitrarily. Operator application is typically constrained by a *search control strategy*. A control strategy specifies which operators should be applied to which of the currently enumerated states. These decisions are based on some notion of the ‘closeness’ of a current state to a goal state or some other ‘quality’ measure. Some strategies treat all states as equally distant from a goal (or equally ‘good’) (e.g., resulting in a depth-first and breadth-first search), while others (e.g., best-first and hill-climbing) use *heuristics* to approximate the distance from a goal. A thorough introduction to state-space search can be found in most general AI texts, including [RICH83, NILS80].

The learning from examples task is easily mapped onto the general search procedure. While there are many possible mappings, one assumes that for every teacher defined class, O_i , a search for a corresponding concept, C_i , is made. C_i must cover all positive (O_i) and no negative ($\neg O_i$) instances. This mapping assumes that a state is a 5-tuple (C, P, N, P', N') : a concept, the set of positive instances (from O_i) covered by the concept, the set of negative instances (from $\neg O_i$) covered by the concept, the set of positive instances not covered by the concept, and the set of negative instances not covered by the concept.

- An initial state is given by $(p_i, \{p_i\}, \{\}, O_i - \{p_i\}, \neg O_i)$, where p_i is a single positive instance that plays two roles: it is the initial concept (recall the single representation assumption) and the only object covered by the concept.
- A goal state is one with a concept, C , that covers all the positive instances, but no negative instances, i.e., $(C, O_i, \{\}, \{\}, \neg O_i)$.
- Many learning operators may exist, but a simple one assumes that a currently uncovered positive instance is used to generate a new concept description from the concept of a current state. The old concept is modified to cover the new instance by “unioning the corresponding value sets of the concept and new positive instance”. For example, if the color of a currently uncovered object is red and all currently covered positive objects are blue, the (partial) effect of this operator is given as

$$\{[\text{Color} = \{\text{blue}\}], \dots \} \implies \{[\text{Color} = \{\text{blue}, \text{red}\}], \dots \}$$

This operator has the effect of forming a more general concept description and is thus called a *generalization* operator.

This is only one way of mapping learning from examples onto state space search. Nonetheless, the mapping is quite general. With the exception of the example generalization operator, the mapping makes no assumptions about the object or concept language. The only major assumption is that the initial state contains a maximally-specific concept and learning moves from specific to more general concepts. An alternative would be to begin with the maximally-general concept and move in the direction of greater specificity [LAN87A]. In some cases this might be achieved by 'inverting' generalization operators and moving 'backward' through the state space,³ but future discussion assumes that if general to specific motion is desired, *specialization* operators will be explicitly supplied. Besides strict unidirectional movement, some systems initiate search at both maximally-general and maximally-specific concepts and move 'inward' [MITC83, MITC77] or search begins at concepts of intermediate generality and moves 'outward' [SCH86c].

There are several factors that influence the cost of searching the space of concepts. Most notably, the concept language defines the search space, thus *biasing* search [MITC83, REND86]. For example, the concept language considered thus far makes important assumptions in this regard: concepts are a *conjunction* of *attribute-value set pairs*. Consider the search traced out in Figure 8 that uses the generalization operator given above. Each new object causes a single new state to be generated.⁴ This 'search' is really deterministic. Allowing disjunction to occur between values of different attributes (e.g., Color = blue \vee Size = large), and not simply internal disjunction, would increase the nondeterminism of search [SCH86c,

³ This is similar to what is done when *backward-chaining* [RICH83, NILS80] from a goal to initial state. The proposal here though is to move from the maximally-general state which is probably not the goal.

⁴ For simplicity, Figure 8 only shows the concepts associated with each state.

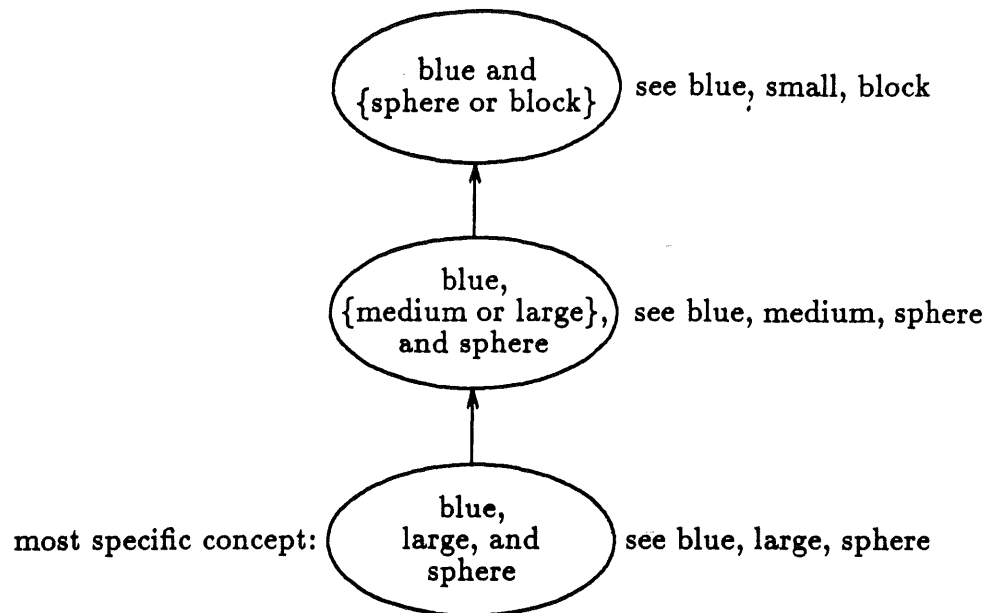


Figure 8

‘Search’ for maximally-specific, conjunctive, attribute-value concepts

MICH81, VER80]. Structured representations also necessitate increased search [MIC83A, VER80, HAYE78, MITC83, PORT84].

Another reason that ‘search’ is deterministic in Figure 8 is that the generalization operator of this example generates a single, maximally-specific concept to accommodate each object. An important result (which is not proved here) of the conjunctive attribute-value assumption is that there is exactly one maximally-specific concept for a given object set. Unioning attribute value sets in the manner above is guaranteed to yield this set. Additional search may also be required if operators are not constrained to generating maximally-specific concepts [MICH80].

2.3.2 The Performance Task

The goal of search in learning from examples is to find a concept that covers all positive and no negative instances. Doing this for each teacher-defined class insures that learned concepts can be used to identify the class membership of all observed objects. However, recall that a concept covers a superset of the positive

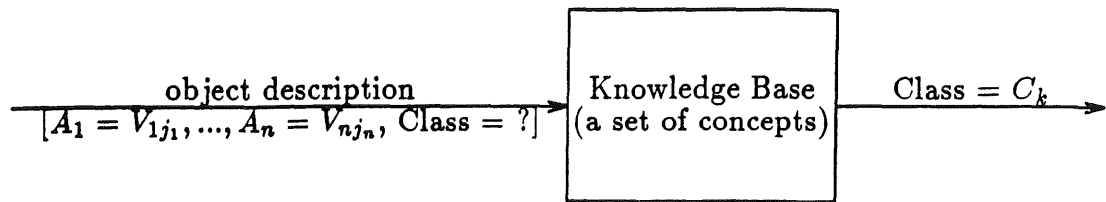


Figure 9

The performance task for learning from examples

instances. Learned concepts can also be used to identify previously unobserved objects with respect to teacher-defined classes. In some cases this identification will be incorrect (as graded by the teacher), but nonetheless, a number of systems demonstrate empirically [QUIN86, HAMP83, SALZ86, SCH86B] or formally [VERE80] that correct prediction occurs a high percentage of the time.

Systems that learn from examples have been applied in many domains, including integral calculus [MITC83, PORT84], soybean diagnosis [MICH81], visual recognition [WINS75], speech recognition [BRAD87], and horse racing [SALZ86]. Despite this diversity, the performance tasks associated with all these systems reduce to identifying objects (states, events, facts, etc.) with respect to teacher defined classes by using induced concepts. Figure 9 demonstrates this task. An object description is matched against a set of previously learned concepts, from which class membership can be predicted.

2.3.3 Summary

Learning from examples is defined by the environmental assumption that objects come preclassified by a teacher. The result is typically a set of concepts, one for each class. Concepts are used to predict the membership of future objects. Appropriate concepts are found through a process of search. Two dimensions of this search process have been alluded to. One is *search control*, which ranges from exhaustive to heuristic strategies. A second dimension is the *direction of*

search, which can proceed from specific to general using generalization operators, or alternatively, may proceed from general to specific.

Learning from examples is a good context for introducing learning as search. However, many situations relax some constraints associated with learning from examples. In particular, conceptual clustering systems do not assume the presence of a teacher. In addition, they organize concepts into hierarchies. These revised assumptions motivate an extension of the search framework, as well as a new performance task. These issues are discussed in the following sections.

2.4 Conceptual Clustering

Conceptual clustering is a process of concept formation that differs from learning from examples in terms of its environment, knowledge base, and performance task assumptions. Briefly stated, conceptual clustering assumes:

- Given:
- A set of objects, O
 - A set of concept descriptions, C , which is usually represented intensionally by a concept language.
- Find:
- Classes or clusters, O_i , that are subsets of O
 - Concepts, C_i in C , that correspond to the classes, O_i .

Objects do not come preclassified as with learning from examples, but object classes must be discovered by the conceptual clustering system. Classes need not be mutually disjoint. Most systems build classes at several levels of generality, thus producing a classification hierarchy. In addition, concepts must be found for each discovered class. This is a subtask that conceptual clustering shares with learning from examples.

2.4.1 The Knowledge Base: Concept Hierarchies

Intuitively, concepts reflect the regularity or structure of the objects classes that they represent. For instance, the concept *has-hair* \wedge *bears-living-young*

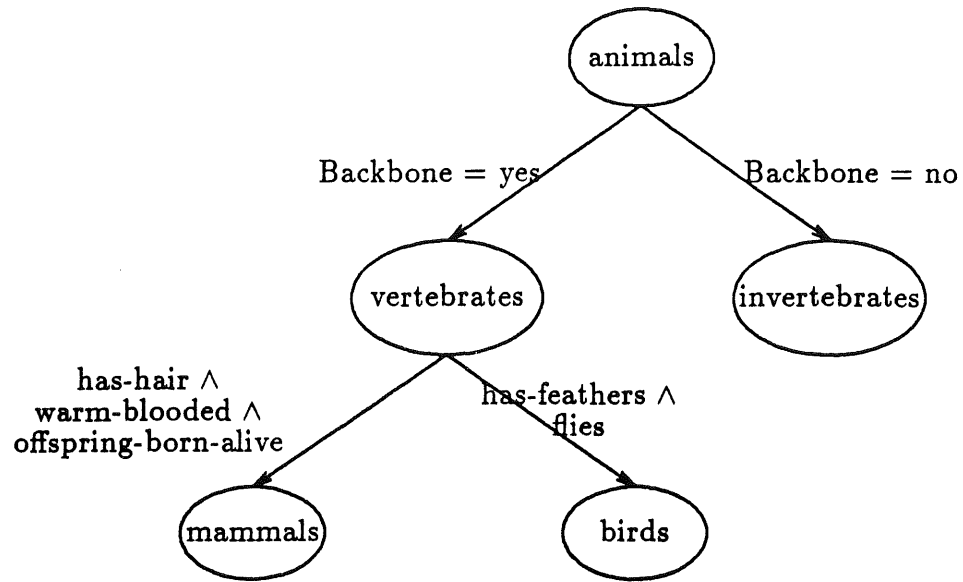


Figure 10

An example decision tree

shows the relation between two properties over the set of mammals. However, for an arbitrary set of objects there may be little commonality, and thus little revealed by the corresponding concept. Therefore, conceptual clustering systems frequently decompose a domain into subsets that can be represented by helpful concepts. A favorite way of structuring this decomposition is a tree.

A popular type of tree is the *decision* or *discrimination* tree [QUIN86, CHAR80, FEIG84, HUNT66]. An example is shown in Figure 10. Each node represents an object class. Depending on convention, concepts or membership *tests* label arcs or nodes of the tree and guide object recognition. For example, an animal with a **backbone** is sifted down the left subtree, where a second test based on Body-cover is made. This continues until a leaf is reached. Leaves typically supply a property (or class name) that can be predicted of objects that have reached that point.

The decision tree of Figure 10 exhibits both *monothetic* and *polythetic* tests. The top node divides objects based on their values along a single attribute (e.g.,

Backbone), and is thus a monothetic classifier. In contrast, vertebrates are partitioned by a polythetic test based on their values along multiple, perhaps differing, attributes [FIS86A, MIC83A]. However, regardless of whether a test is monothetic or polythetic, each is a concept that discriminates objects of one class from all others and for that reason is termed a *discriminant* concept [MIC83C].

Although decision trees (and the processing assumptions typically associated with them) are of great utility, they have weaknesses. First, they are not well-suited for recognizing objects with missing information. If an object is encountered that does not exhibit an attribute value(s) required for a test, then recognition cannot proceed [BARS84], other than to explore all possible subtrees [QUI87B]. Second, predictions are limited to the leaves of the tree. Intermediate predictions may be useful when partial information does not allow traversal to a leaf. Last, decision trees do not allow exceptions of arc tests to be placed in the same class. For example, suppose $\text{has-hair} \wedge \text{warm-blooded} \wedge \text{offspring-born-alive}$ is a discriminant concept for mammals. However, because platypi lay eggs they cannot be placed with other mammals (unless offspring-born-alive is removed as a discriminant feature of mammals).

Motivated by these restrictions, several variants of the decision tree have evolved [LEBO82, KOL83A, KOL83B]. The Generalization Based Memory (GBM) organization of UNIMEM [LEBO82] exemplifies how problems with decision trees can be overcome. First, GBM indexes nodes with multiple tests. For example, a node corresponding to mammals might be indexed by three *different* arcs corresponding to has-hair, offspring-born-alive, and warm-blooded. Thus, recognition can proceed when some properties are unknown. Second, GBM distinguishes arc-labeling concepts that discriminate objects from node-labeling concepts. The latter concepts give properties common to all class members and are called *characteristic* concepts [MIC83C]. Arc-labeling concepts correspond to sufficient conditions and

node labeling concepts are necessary conditions for class membership. Of course, some conditions may be necessary and sufficient and thus be used at both nodes and arcs. The important ramification of using node concepts is that intermediate predictions can be made. For example, having reached the mammal 'node' (perhaps by virtue of observing hair) suggests that the animal is also warm-blooded. Last, while early versions of GBM did not allow exceptions to be introduced, later versions have done so [LEBO87]. Exceptions mean that node conditions are no longer necessary, but are only *default* values. More will be said later about the nonmonotonic reasoning that arises when using default values [BRAC85, ETHE83, REIT80]. Additionally, GBM relaxes the sufficiency assumption by allowing attribute values to index more than one concept.

An example GBM structure is shown in Figure 11. The important extensions to the basic decision tree structure are the multiple indexing of nodes, node-labeling concepts that are a conjunction of typical properties, and exceptions. This last extension, along with relaxation of the sufficiency condition on arcs, introduces some nondeterminism into the recognition process; concepts no longer logically partition objects, but a process analogous to evidence combination and partial matching [RICH83] must be used to guide recognition.

2.4.2 The Learning Element

Conceptual clustering is a process abstraction originally defined by Michalski [MICH80]. Conceptual clustering does not assume that objects are preclassified. Instead, a system must discover classes for itself. Most systems arrange these classes into a hierarchy. This process is not haphazard, but is guided by quality measures that rank classes and concepts. This section focuses on the mechanisms that underlie many conceptual clustering systems. This entails a detailed examination of several systems. More generally, conceptual clustering will be viewed as

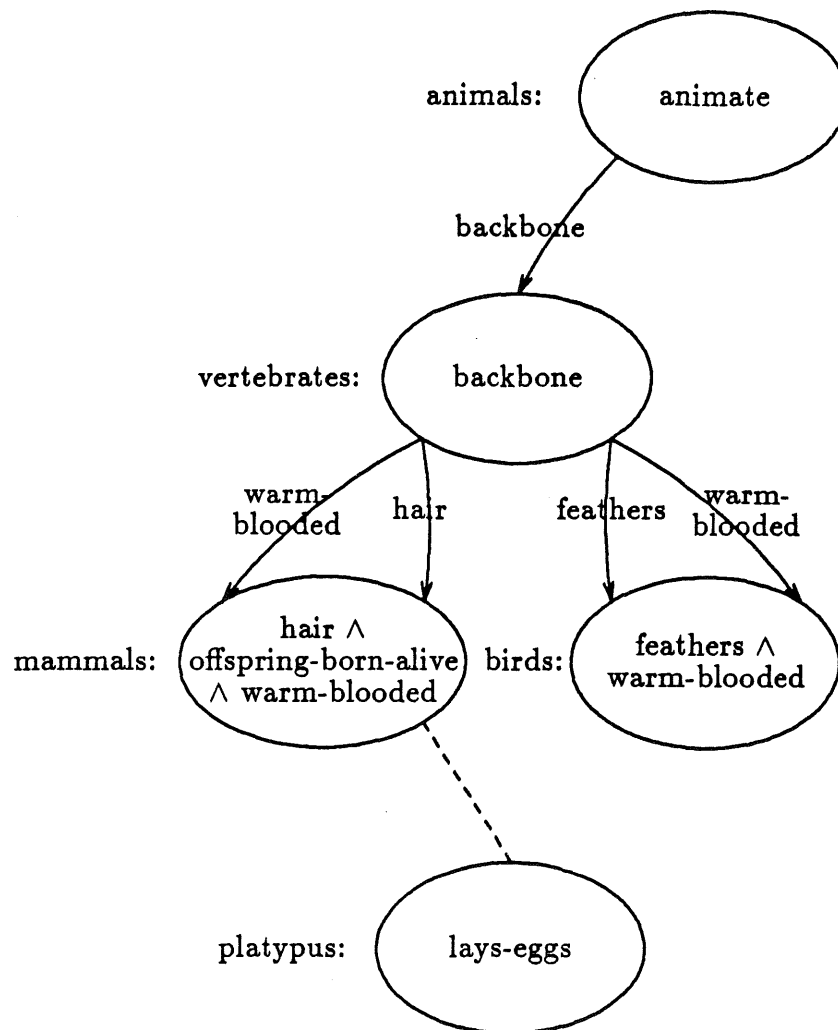


Figure 11

An example of GBM

a process of search. However, conceptual clustering will first be distinguished from a class of methods known as *numerical taxonomy*.

2.4.2.1 Conceptual Clustering and Numerical Taxonomy

Michalski proposed conceptual clustering as an alternative to numerical taxonomy [EVER80], a class of techniques developed by social and natural scientists to analyze experimental and observational data. Like conceptual clustering, these methods form classification schemes. Unlike conceptual clustering however, they

do no form concept descriptions for discovered classes. Specifically, Michalski [MICH80] distinguishes conceptual clustering and numerical taxonomy by the way each evaluates the quality of object classes.

In methods of numerical taxonomy [EVER80], the similarity between two objects is the value of a function applied to the object descriptions. The description of an object is much like the attribute-value representation described earlier, but typically similarity is defined around the assumption that numeric attributes predominate. Typically, a data analyst computes the pair-wise similarity of all objects in a data set and inputs a matrix of these similarities to a numerical taxonomy program. The similarity matrix is used to group objects that are most similar and to distinguish objects that are least similar. Intra-cluster and inter-cluster similarity are computed by a function of the pair-wise similarities of the objects in each cluster. Given two objects, A and B , with descriptions, A' and B' , Michalski [MICH80] points out that a typical similarity measure between A and B has the form

$$\text{Similarity}(A, B) = f(A', B').$$

This measure is *context-free*, since the similarity between A and B is independent of A 's and B 's relationship to other objects being clustered. *Context-sensitive* measures of similarity have also been developed in which the similarity of two objects is dependent on their relation to additional objects. That is, within a set of objects, O , with a set of symbolic descriptions, O' , the similarity of two objects, A and B , has the form

$$\text{Similarity}(A, B) = f(A', B', O').$$

If we assume integers are 'objects' then using a context-sensitive measure would result in greater similarity between the integers 1 and 9 when considered within the range 1 to 100 than when considered within the range 1 to 10.

Using a numerical taxonomy program, the data analyst guides the search for useful classification schemes by standardizing the raw data in various ways and/or by using different similarity functions to build the similarity matrix input to the program. However, despite the usefulness of numerical taxonomy techniques, any such method (whether it uses context-free or context-sensitive measures) suffers from a major limitation - the resultant clusters may not be easily characterized in a generalized conceptual language. This limitation can be of concern to a data analyst (or learning program) who wants to abstract the underlying conceptual structure of object clusters in order to hypothesize about future observations. In conceptual clustering, a cluster is not simply an extensional enumeration of objects, but is intensionally represented by a concept.

Conceptual clustering addresses the problem of determining conceptual representations of object clusters. Given a set of concepts, C , Michalski defines the similarity between two objects, A and B , as

$$\text{Similarity}(A, B) = f(A', B', O', C).$$

In other words, the similarity between two objects is dependent on the quality of concepts used to describe them. Extending this idea, the quality of an object cluster is dependent on the quality of the concept that describes it.

Michalski and Stepp [MIC83A, MIC83B] suggest several quality measures for conceptual clustering that are a function of concepts or the map between concepts and the objects they cover. For example, assuming the attribute-value representations described earlier, *simplicity* is a function of the size and number of value sets present in a concept after dropping conditions. The simplest concept is null

(maximally general). On the other hand, *fit* measures the ratio between the number of observed members of a concept and the number of possible objects covered by the concept. The 'tightest-fitting' concept for a set of observations will be the most specific concept that covers them.

In practice, Michalski and Stepp measure class quality as a tradeoff between simplicity and fit (and possibly other measures). The rationale for these measures is that simple and 'tight-fitting' concepts are easier to understand, and classes so described should be favored. The tradeoff between simplicity and fit is analogous to the numerical taxonomy tradeoff between inter- and intra- cluster difference. However, in conceptual clustering this give and take occurs at the level of concept, not object, descriptions.

2.4.2.2 Types of Conceptual Clustering Systems

Conceptual clustering systems differ along many dimensions. However, systems can be initially distinguished by the types of classification schemes that they form, a dimension frequently used to classify numerical taxonomy methods. Discussion to follow gives a complete account of methods to date, some of which will be pursued in more detail later.

Optimization techniques form a 'flat' (i.e., unstructured) set of mutually exclusive classes and concepts (as with learning from examples). Optimization techniques make an explicit search for a globally optimal K-partition of an object set, where K is a user-supplied parameter. This search makes optimization techniques computationally expensive, thus constraining their use to small data sets and/or small values of K. An example of an optimization method is the partitioning module of Michalski and Stepp's CLUSTER/2 system [MIC83A, MIC83B].

Hierarchical techniques form classification trees over object sets, where leaves are individual objects and internal nodes represent object clusters. Decision trees

are the predominant representation used by such systems. A 'flat' set of mutually-exclusive clusters may be obtained by severing the tree at any level. Hierarchical techniques depend on 'good' clusterings arising from a series of 'local' decisions. The use of 'local' decision-making in hierarchical methods make them computationally less expensive than optimization techniques, with a possible reduction in the quality of constructed clusterings.

Hierarchical techniques are further divided into *divisive* and *agglomerative* methods, which construct classification trees top-down and bottom-up, respectively. Agglomerative methods begin by assuming that each object is its own singleton class. Classes are successively grouped until a single, all-inclusive class is achieved. Figure 12 shows example 'snapshots' of an agglomerative method at work - concepts for the object classes are not shown. In this example, $\{o_1\}$ and $\{o_2\}$ are initially the most similar classes by a similarity measure that will not be specified here. After merging them, $\{o_1, o_2\}$, $\{o_3\}$, $\{o_4\}$, and $\{o_5\}$ remain. Of these, $\{o_4\}$ and $\{o_5\}$ are the most similar. Combining these singletons results in a partition $\{\{o_1, o_2\}, \{o_3\}, \{o_4, o_5\}\}$. The singleton $\{o_3\}$ is then merged with $\{o_1, o_2\}$. Finally, the last snapshot shows that the remaining classes, $\{o_1, o_2, o_3\}$ and $\{o_4, o_5\}$, are combined into a single class.

The example of Figure 12 also demonstrates why hierarchical methods are less expensive than optimization methods. 'Local' decisions that grouped o_1 and o_2 , and o_4 and o_5 , precluded the possibility of ever forming $\{o_1, o_2, o_4\}$. An optimization technique would consider this possibility and many others. Agglomerative methods include work by Cheng and Fu [CHEN85], MK10 by Wolff [WOLF80], and WITT by Hanson and Bauer [HANS86].

In contrast to agglomerative systems, divisive methods begin with a single, all-inclusive class and continually divide classes until singletons are reached or some

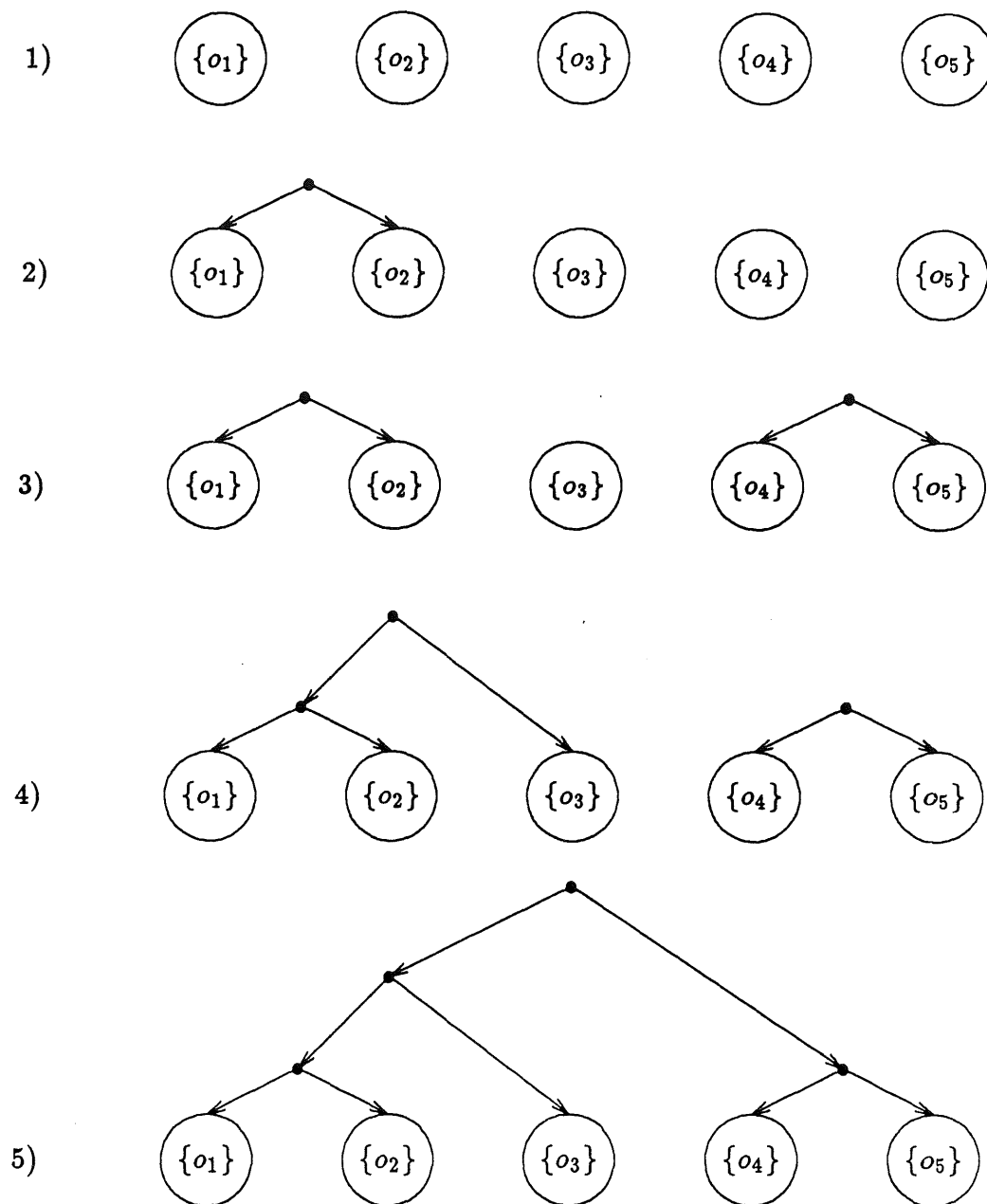


Figure 12

An agglomerative method at work

other termination condition is satisfied. Divisive systems include the Hierarchy-building module of CLUSTER/2 by Michalski and Stepp [Mic83A, Mic83B], Stepp's CLUSTER/S [STEP84, STEP86], DISCON by Langley and Sage [LANG84], Fisher's

RUMMAGE [FIS85B], and OPUS by Nordhausen [NORD86]. A particularly interesting method is the Hierarchy-building module of CLUSTER/2; it calls an optimization technique (the partitioning module) to divide classes. By building the tree with a small branching factor (e.g., 2), a K-partition of all objects can be cheaply obtained by appropriately severing the tree. However, this method is likely to result in a partition of lesser quality than one obtained by calling the partition module directly for partition size K.

Clumping techniques build classifications where classes may overlap. The possibility of class overlap stems from independently treating some number of classes as possible hosts. In hierarchical and optimization techniques an object is placed in the class that maximizes a quality function. In clumping techniques a class is selected if it satisfies some constraint that is independent of other classes (e.g., a quality score surpasses a threshold). GLAUBER by Langley, Zytow, Simon, Bradshaw, and Fisher [LAN86B, LANG85], and NGLAUBER by Jones [JONE86] were originally framed as conceptual clumping methods. In addition, Lebowitz' UNIMEM [LEBO86, LEBO85, LEBO82] has been so framed by Fisher and Langley [FIS86A] and Kolodner's CYRUS [KOL83A, KOL83B] can be similarly described. All of these techniques form classification hierarchies where objects may be classified in several places and nodes may have multiple parents. UNIMEM's GBM [LEBO82] allows clumps, although discussion in section 2.4.1 did not consider the clumping aspect of this work. Like the hierarchical methods above, clumping methods may be distinguished in terms of top-down versus bottom-up processing.

Despite variety in the classification schemes formed by conceptual clustering systems, there is a good deal of commonality in their processing. The next section looks at three systems in more detail.

2.4.2.3 Selected Conceptual Clustering Systems

Both learning from examples and conceptual clustering are concerned with formulating descriptions that summarize data. In the former a teacher dictates class assignment, leaving the learner to characterize each class. In the latter the learner has the two-fold task of creating object classes as well as characterizing these classes. Thus, there are two problems that are addressed by a conceptual clustering system:

- The *clustering problem* is the problem of partitioning a set of objects into subsets. In other words, extensional object classes are formed. The clustering problem is addressed by conceptual clustering and numerical taxonomy, but not by learning from examples.
- The *characterization problem* is the problem of determining concepts for extensionally represented object classes. This is simply the problem of learning from examples. This problem is addressed by conceptual clustering and learning from examples, but not by numerical taxonomy.

Given this view, a natural approach in conceptual clustering is to form object classes and then use traditional methods of learning from examples to find concepts. These concepts can be used to evaluate the quality of the classes they represent. In fact, most present conceptual clustering algorithms can be framed this way.

RUMMAGE and DISCON

RUMMAGE [Fis85B] and DISCON [LANG84] both use a simple solution to the clustering problem. Each assumes objects are represented as attribute - value pairs. The values of an attribute collectively imply a partition of an object set, where objects with the same value are members of the same class. RUMMAGE solves the clustering problem by considering a number of partitions, each implied by the values of a distinct attribute. It selects that partition with the 'best' conceptual descriptions over the remaining attributes, thus solving the characterization problem. RUMMAGE evaluates concept quality using variants of Michalski and Stepp's measures of simplicity and fit. Since it selects partitions based on the

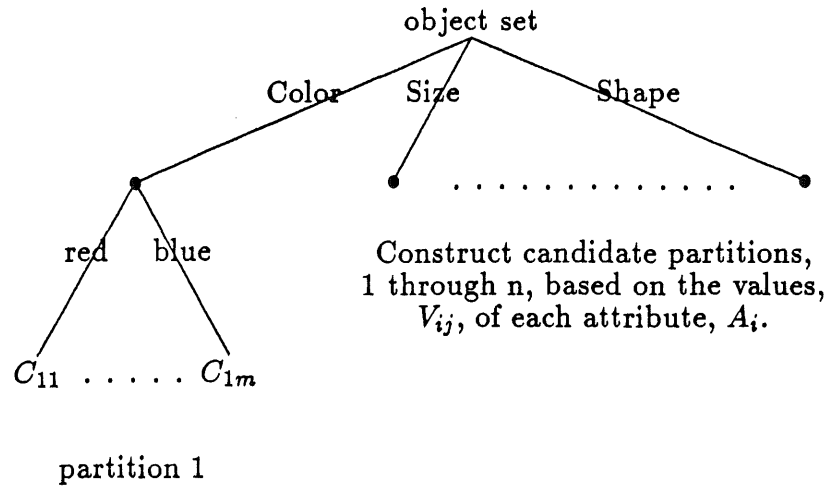


Figure 13

Clustering in RUMMAGE and DISCON

values of a single attribute, RUMMAGE forms monothetic trees. RUMMAGE applies this method recursively to each of the resulting clusters, tracing out a single classification tree. In effect, RUMMAGE produces a classification tree where nodes have 'good' concept descriptions over attributes that have not been used previously as classification tests. Like RUMMAGE, DISCON uses attribute values to imply possible partitions. However, it does not construct an explicit description of the devised clusters, but simply calls itself recursively on each of the possible clusters, thus forming a classification tree over the objects of each cluster with respect to the remaining attributes. The attribute that implies subtrees with the least number of total nodes is chosen to initially divide the object set. As a result of applying this procedure recursively DISCON finds a classification tree with the least number of nodes. That is, DISCON finds the tree that on average classifies the observed objects in the least number of tests. Both RUMMAGE and DISCON are loosely based on Quinlan's ID3 program [QUIN86, QUIN83]. An example of the clustering processes of both RUMMAGE and DISCON is given in Figure 13.

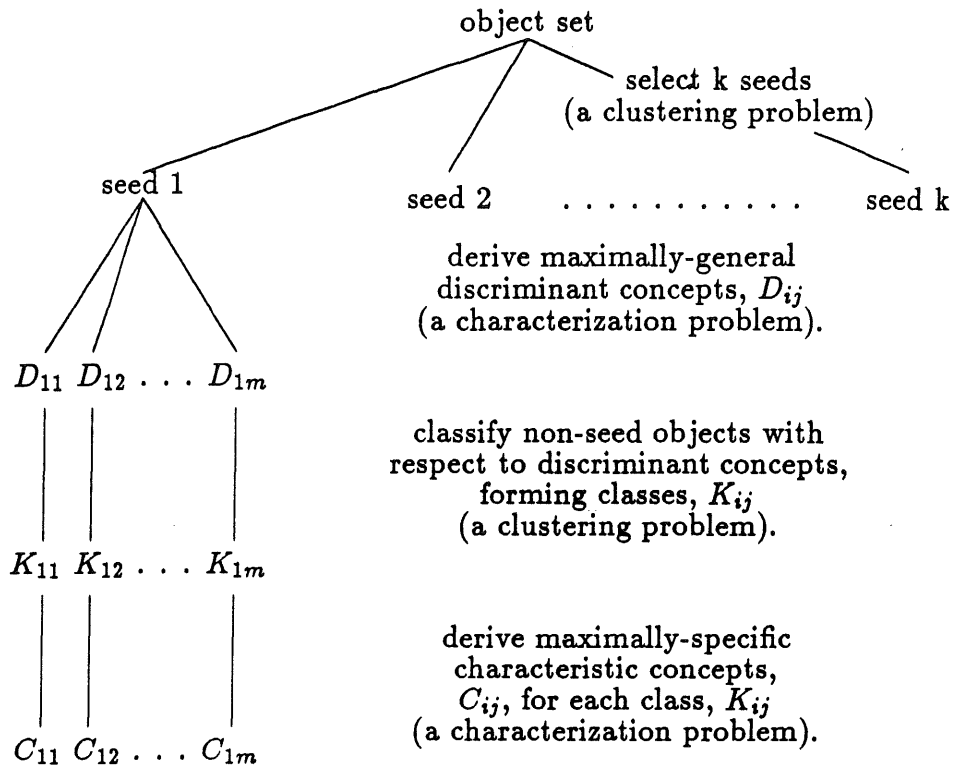


Figure 14

Processing in the Partitioning Module of CLUSTER/2

The Partitioning Module of CLUSTER/2

The Partitioning Module of Michalski and Stepp's CLUSTER/2 system [MIC83A, MIC83B] uses a more experimental solution to the clustering problem than RUMMAGE or DISCON. Given the task of dividing the observed objects into N disjoint classes, the system initially selects N *seed* objects (initially this is done randomly). The system treats each seed as a positive instance of some class and treats the other seeds as negative instances of the same class. The program then derives *maximally-general discriminant concepts* for each class (each class is a singleton). As the name implies, a maximally-general discriminant concept is the most general concept that discriminates the positive instances from all negative instances. The result is that for each seed a number of concepts are derived, each

of which covers that seed and no other seed. Each concept also covers some number of non-seed objects. Once all objects (seed and non-seed) have been classified with respect to the maximally-general discriminant concepts, these concepts are 'thrown out', and maximally-specific concepts are derived for each class. By selecting one concept for each seed, a set of (possibly overlapping) clusters results that classifies the input object set. A pictorial summary of this process is shown in Figure 14.

The reasons for this seemingly roundabout means of clustering and characterization are best explicated in Michalski [MICH80]. By first formulating maximally-general descriptions, *any* clustering implied by any combination of maximally-general descriptions (one description for each seed) can be shown to contain *at least one* cluster that covers an arbitrary object. By first formulating maximally-general descriptions, CLUSTER/2 guarantees that every observed object is classified. Once all objects are classified, maximally-specific descriptions reduce the possibility of overlapping clusters with respect to unobserved objects. A 'fix-up' operation is then employed to make all possible clusterings mutually-disjoint. In general, each concept is composed of multiple attributes. Therefore, CLUSTER/2 forms polythetic rules.

2.4.2.4 Conceptual Clustering as Search

Descriptions of individual systems show that clustering and characterization are not independent. In fact, the definition of conceptual clustering implies the results of characterization (i.e., a set of concepts) must be used to determine the quality of object clusters (i.e., the result of clustering). A learning from examples (characterization) task is embedded within a larger task of forming object clusters (clustering). Fisher and Langley [FIS85A, FIS86A] adapt the view of learning as search to fit conceptual clustering. Clustering and characterization dictate a two-tiered search, a search through a space of object clusters and a subordinate search through a space of concepts. This is demonstrated by Figures 16 and

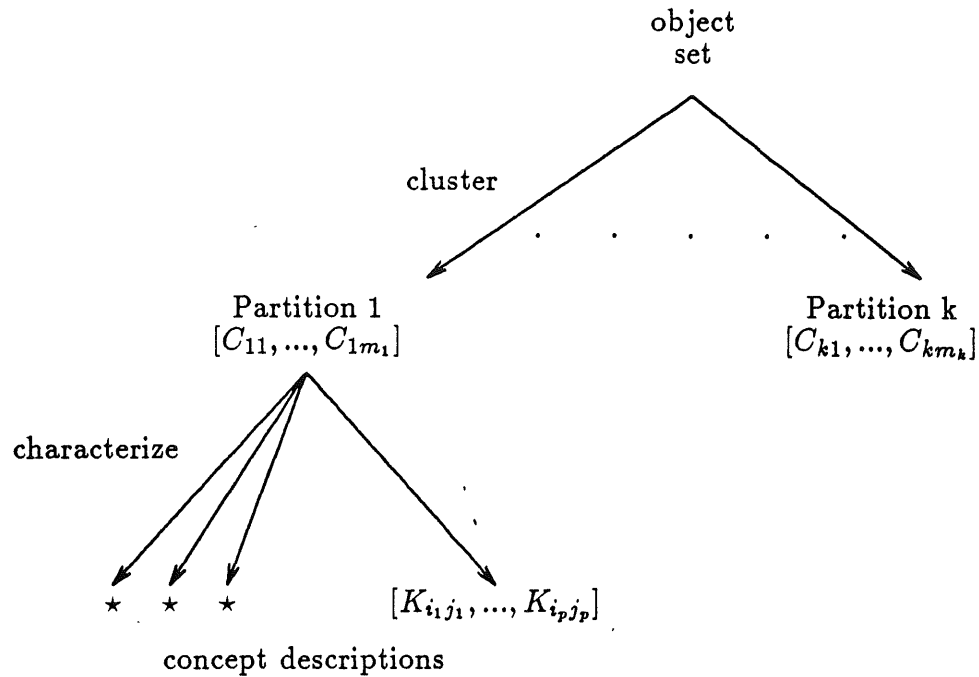


Figure 15

Generating clusterings in conceptual clustering

17, which depict the expansion and evaluation phases of search. In the case of hierarchical techniques this becomes a three-tiered search, with a top-level search through a space of hierarchies. The snapshots of Figure 13 (i.e., the example of an agglomerative method) give an intuition as to the states in this space.

Just as dimensions can be applied to the characterization search process in learning from examples, dimensions such as search control and direction distinguish conceptual clustering systems at any of the three levels of search. For instance, most systems transform a single classification tree throughout processing and thus perform hill climbing through the space of hierarchies (e.g., CLUSTER/2 [MIC83A, MIC83B] and RUMMAGE [FIS85B]). On the other hand, Langley and Sage's DISCON system [LANG84] makes a nearly exhaustive search of hierarchy space, and selects that classification with the fewest nodes. Second, when searching

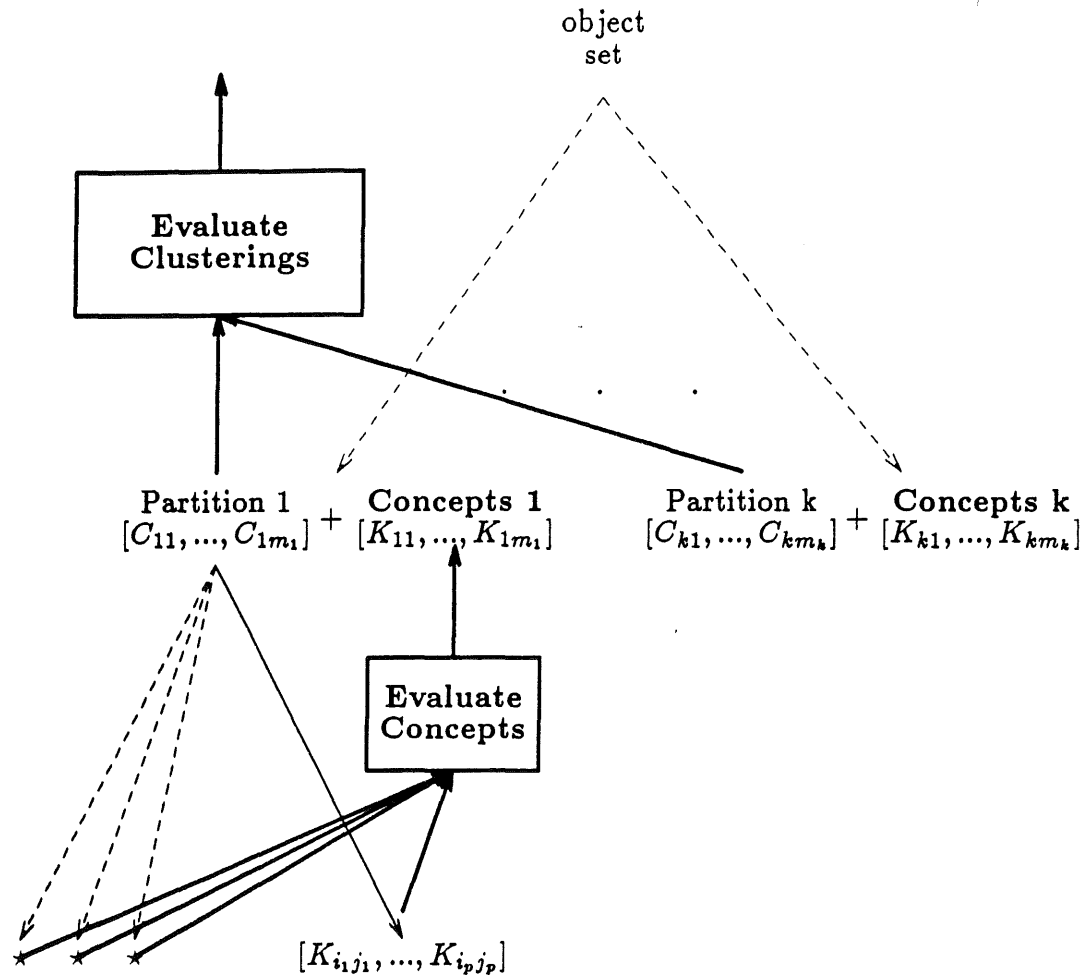


Figure 16

Evaluating clustering in conceptual clustering

through a space of hierarchies, search direction may dictate building a tree top-down (divisive techniques) by continually dividing nodes [LANG84, MIC83A, FIS85B] or building a tree bottom-up (agglomerative methods) by continually fusing nodes [WOLF80, HANS86, CHEN85]. Although there is variety in searching hierarchies, most systems search for maximally-specific, conjunctive, attribute-value representations at the level of characterization. Recall that under this assumption the characterization process is deterministic. Exceptions include CLUSTER/2 that

searches for more general concepts and CLUSTER/S by Stepp [STEP84, STEP86], that clusters structured objects.

One interaction between the search for characterizations and the search for partitions has been mentioned; the result of characterization must be used to evaluate competing partitions. More generally, the nondeterminism of a subordinate search constrains the possible nondeterminism of higher level search. For example, in searching a space of partitions there are two sources of nondeterminism: the number of generated partitions and the number of characterizations produced for each cluster (for simplicity, Figure 16 showed only 1 characterization being returned). In the case where only one concept for each cluster is being returned (e.g., when searching for maximally-specific, attribute-value concepts), the number of characterizations is eliminated as a source of nondeterminism in searching partitions.

The nondeterminism of searching hierarchy space is similarly constrained by the search for partitions. The number of trees generated cannot exceed the number of partitions being investigated in the subordinate search. For example, the hierarchy building module of CLUSTER/2 performs a beam search (of size M) through the space of partitions, but maintains only a single tree (beam size = $1 \leq M$), and therefore is hill climbing through this space. On the other hand, DISCON investigates all partitions and all trees as well.

The search framework clarifies the mechanisms of conceptual clustering. This clarity makes the impact of environmental and knowledge base assumptions on processing more accessible. The next section drops the environmental assumption that all objects are present at once. A strategy for incremental conceptual clustering is developed that is distinguished from nonincremental methods by search direction and control strategies.

2.5 Incremental Concept Induction

Many concept learning systems, whether they be of the learning from examples or conceptual clustering variety, are *nonincremental*, in that all objects over which induction occurs must be simultaneously accessible to the system. In contrast, *incremental* methods accept a stream of objects that are assimilated one at a time. A primary motivation for using incremental systems is that a knowledge base may be updated as each new object is observed, thus sustaining a continual basis for reacting to new stimuli. This property is paramount if systems are to be used in real-world environments [CARB86, LAN86A, SAMM86]. Therefore, search-intensive methods (e.g., depth-first, breadth-first, best-first search) may not be appropriate in incremental systems, since they require updating a frontier of concept hypotheses and/or examining a list of previously seen objects. For this reason, a profitable view is that incremental strategies operate under diminished search control. Specifically, this supposition is investigated by looking at some existing concept learning programs. Each system uses hill climbing to keep object assimilation costs down. They also include mechanisms that maintain learning robustness.

2.5.1 Incremental Learning from Examples

Systems that learn from examples typically conduct a single-level search through a space of characterizations or concepts. Two systems are described that exemplify the hill-climbing strategy for incremental learning from examples.

'ARCH'

Winston's 'ARCH' [WINS75] learns conjunctive structural descriptions from examples. Consider the positive and negative instances of an arch in Figure 17. The system's initial concept for the positive instances is simply a positive instance. Representing the positive instance of Figure 17 as a conjunction of predicates yields

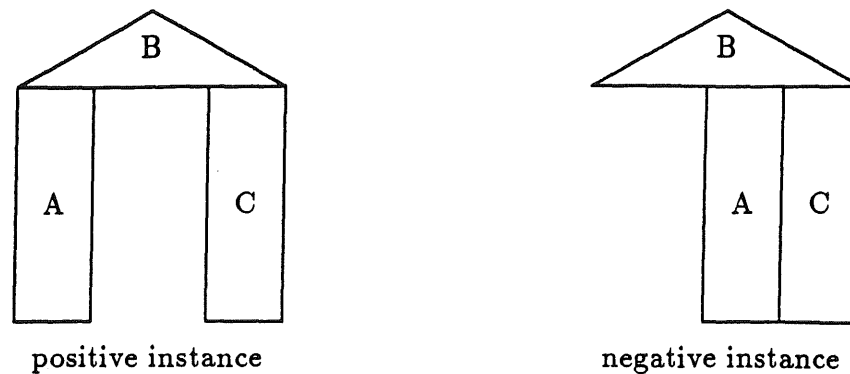


Figure 17

Examples for the 'ARCH' program

$$\text{Shape}(B, \text{wedge}) \wedge \text{Shape}(A, \text{block}) \wedge \text{Shape}(C, \text{block}) \wedge \\ \text{Ontop}(A, B) \wedge \text{Ontop}(A, C)$$

Suppose that a second positive instance is observed where the top component is a block, not a wedge. 'ARCH' matches the description of this instance with the current concept and a generalized concept description is formed to cover the new instance.⁵ The new concept is

$$\text{Shape}(A, \text{block}) \wedge \text{Shape}(C, \text{block}) \wedge \\ \text{Ontop}(A, B) \wedge \text{Ontop}(A, C)$$

Eliminating a Shape predicate indicates that one of the components may be any shape. Next the negative instance of Figure 17 is given by the 'teacher'. This instance matches the current concept description for positive instances. To make the concept consistent it must be specialized. There are many ways to specialize this concept and achieve consistency. Here are three: (1) add a predicate, $\text{not-Touching}(A, C)$, to the conjunction, (2) add a predicate, $\text{At-end-of}(A, B)$, or (3) add a predicate, $\text{not-Shape}(B, \text{wedge})$.⁶

⁵ The matching process for structured descriptions will be left to the reader's intuition, but in general there can be several possible ways to do the match, one of which is selected by 'ARCH'.

⁶ Winston uses 'near misses' - negative instances that differ only in one respect to the current concept description - to limit the number of possible specializations

The last alternative, though making the concept consistent (with respect to the negative instance), makes it incomplete as well, since the first positive instance is no longer covered. However, 'ARCH' does not save instances and has no way of knowing this. Vere [VERE80] first pointed out this weakness of 'ARCH'. The second alternative maintains completeness while insuring consistency. However, it is simple to create an arch where both supports are slightly indented. The second alternative may cause problems down the road that will require generalization, possibly even a return to the concept just specialized. Only option (1) will satisfy current and past knowledge, and not unnecessarily complicate the incorporation of future instances. Of course, the system has no way of knowing or using this when selecting one (and only one) option.

'ARCH' can be viewed as a hill climber since it considers only one hypothesis at a time. In addition, the system can generalize or specialize its concept description through operator application. Rather than moving in only one direction, 'ARCH' has bidirectional mobility through the the space of concept descriptions. While this strategy is computationally cheap, it can also lead to incorrect hypotheses and there is the possibility of cycling between states.⁷

ID4

Schlimmer and Fisher [SCH86A] have developed a number of incremental variants of Quinlan's ID3 [QUIN86, QUIN83]. Unlike most systems that learn from examples, ID3 forms a decision tree. An object is sifted down the tree until a leaf

that need be considered. However, present discussion considers such an assumption external to the learning from examples system.

⁷ The system can also backtrack if a *dead-end* is reached; this is a concept description in which there is an explicit contradiction among predicates (e.g., Touching(A, C) and not-Touching(A, C)) in the current hypothesis. Thus, 'ARCH' has two ways of 'moving bidirectionally'. One is through backtracking which is part of the system's search control strategy and the second is by using operators that specialize and generalize. Future discussion will be concerned only with the latter meaning.

is reached. Every leaf is labeled by a class designation that is used to predict the membership of a recognized object.

An ID3 decision tree is monothetic. An object set is divided by the values of the attribute that maximizes an information theoretic measure; knowing the value of this attribute is most helpful (among all attributes) in predicting the class of an object. After dividing the top-most node of the tree, division of subnodes continues recursively until it is not useful to do so anymore. This happens when no attribute conveys significant information about class membership. A special case of this occurs when all objects at a subnode are members of the same class. In general, the chi-square measure of statistical independence is used to stop further expansion of the tree.

ID3 is nonincremental. It requires simultaneous access to all objects in order to compute the divisive, information-theoretic heuristic and the chi-square measure. On the other hand, Schlimmer and Fisher's variant, ID4, builds a decision tree incrementally. Beginning with an initially 'empty' tree, ID4 records important statistical information as objects are observed. It does not save instances, but only retains summary statistics. When statistics indicate that a node division is reasonable (according to chi-square), it is done along the 'best' attribute by ID3's divisive heuristic; 'empty' subtrees are created as children. Subsequent objects cause statistics at the root to be updated. Objects are passed down to the appropriate subtree and dictate changes in subnode statistics as well. This process continues indefinitely.

Because of its incremental nature, ID4 may initially choose an attribute for node division that later proves to be nonoptimal according to the divisive heuristic. That is, as statistics change to reflect a growing body of observations, they favor a different divisive attribute. When this happens ID4 throws out the subtrees rooted

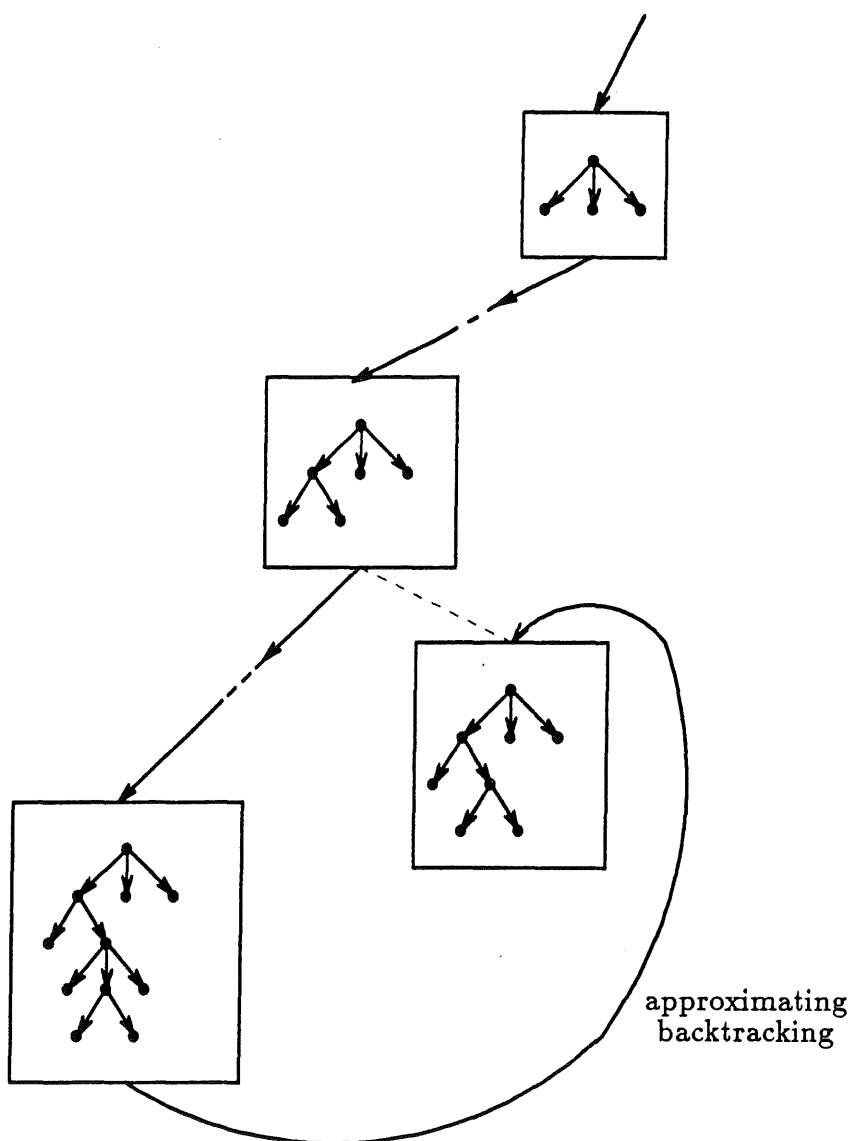


Figure 18

Approximating backtracking by 'forgetting' in ID4

at the offending node and creates a new set of 'empty' subtrees that correspond to the values of the newly determined 'best' attribute.

Unlike most learning from examples systems, ID4 and its predecessor ID3, can be viewed as searching a space of decision trees. However, as an incremental learner, ID4 demonstrates some of the same characteristics as 'ARCH'. ID4 hill

climbs through the space of decision trees. It builds a tree by repeated node division, but can also undo the effects of past expansion by dropping subtrees. This approximates backtracking without the associated overhead of saving past states or instances Figure 18 illustrates the process of approximating backtracking in ID4.⁸ As with 'ARCH', this process is inexpensive, but can lead to cycling. Dropping subtrees also means that learning over a subset of past instances must be repeated. For this reason ID4 may require more instances than ID3 to converge on the same decision tree.

Schlimmer and Fisher identify three dimensions for evaluating any incremental system:

- The *cost of updating* a knowledge base to accommodate a new instance.
- The *number of observations* required by a concept learning system to obtain a 'stable' knowledge base.
- The *quality of concept descriptions* obtained by a concept learning system.

They show that ID4 cheaply integrates new observations into its decision tree. This property stems from the hill-climbing control strategy. Additionally, ID4 converges on decision trees that correctly predict class membership to the same degree as the nonincremental ID3. Quality is maintained by approximating the effects of backtracking by operator application. This approximation ability is similar in intent to *dependency-directed backtracking* [RICH83] used by some nonmonotonic reasoning systems.⁹ Last, in general ID4 required more observations to find the desired tree than did ID3.

2.5.2 Incremental Conceptual Clustering

A hill-climbing control strategy, paired with bidirectional operator mobility, is effective for incremental learning from examples. However, this strategy can also be extended to conceptual clustering. This section describes a system that can be

⁸ Unlike, 'ARCH' which allows backtracking proper, ID4 is completely dependent on operator application to undo the effects of past learning.

⁹ An observation due to Jeff Schlimmer.

viewed as an incremental method of conceptual clustering. Langley, Gennari, and Iba [LAN87B] discuss further examples of this approach to concept formation.

UNIMEM

Lebowitz' UNIMEM [LEBO86, LEBO85, LEB83A] was not originally framed as a conceptual clustering system. It was abstracted from the earlier IPP system [LEB83B], which integrated news stories on international terrorism into a knowledge base. However, UNIMEM has been framed as a conceptual clustering system by Fisher and Langley [FIS86A, FIS85A].¹⁰

To be more specific, UNIMEM carries out conceptual clumping; it builds classification hierarchies that allow objects to be classified under multiple nodes. Given a new object and an existing hierarchy (GBM structure) that was built from previous observations, UNIMEM incorporates the object into the hierarchy. This is a two-step process. Starting at the root of GBM, arc-labeling values that are shared by the object 'activate' some number of subnodes. Arc-labeling values are *predictive* of object classes and are used to make some initial guesses as to what nodes will match the object. Node-labeling values of each activated node are then compared to the attribute values of the object. If the object's values do not disagree with any node labeling value, the object is recursively classified under the node. Node-labeling values are *predictable* of all class members and must be satisfied by the new object. Notice that the success of an object-node match is independent of whether the object matches other activated nodes – thus UNIMEM's clumping behavior.

During classification UNIMEM can alter the classification hierarchy. If an object matches enough (according to a user supplied threshold) of an activated

¹⁰ At the suggestion of Dennis Kibler.

node's predictable values then the object and node can be generalized. A superordinate node is created with predictable values that are the intersection of the object and old node's predictable values. This is similar to the operation of agglomerative techniques, but it only considers object-node combinations. In general, agglomerative methods allow arbitrary nodes to be combined.

There are other ways to modify GBM during classification. Each predictable value of a node has an associated integer count. If a node is activated by an object but has a predictable value that violates the object (e.g., Color = red versus Color = blue), the count(s) of the unmatched value(s) is decremented. In all cases, if a predictable value matches an object value, its count is incremented. If a predictable value's count falls below a user supplied threshold, the value is removed as a predictable value of the node. This is a second form of generalization allowed by UNIMEM. If the number of predictable values falls below a user supplied threshold, the node and all of its subtrees are thrown away. Analogous update rules are followed for predictive values.

Object incorporation results in a single new GBM structure that covers the new object as well as previously classified objects. Since UNIMEM only maintains one hierarchy following each observation, it can be viewed as hill climbing through a space of hierarchical classifications. It bears a limited similarity to agglomerative conceptual clustering methods in that objects can be combined with existing nodes to form higher-level nodes. Secondly, UNIMEM can drop subtrees as a way of undoing previous generalizations based on changing statistics. In this respect, its control structure is similar to that of ID4.¹¹ UNIMEM is also similar to Kolodner's CYRUS system [KOL83A]. In fact, at the level of abstraction that UNIMEM has been described here, CYRUS is nearly identical. Differences between these systems will be explained as they become relevant.

¹¹ Given the chronology, it is more accurate to say that ID4 is similar to UNIMEM.

While existing descriptions of UNIMEM are oblivious to search concerns, desirable search properties can be abstracted from them. Diminished search control and greater operator flexibility is a practical strategy for incremental learning. However, there has been little evaluation of UNIMEM along the obvious dimensions suggested by this framework: update cost, convergence time, and hierarchy quality. Though, Lebowitz [LEBO87] has recently addressed these issues.

2.5.3 Summary

The premise of this section is that incremental processing demands rapid knowledge base update. This need drives the selection of learning strategies. Several incremental learning systems have been examined and a common search methodology has been abstracted. Each uses a hill-climbing control strategy and operators that allow search to progress bidirectionally in appropriate state spaces. This abstraction has not been explicit in previous descriptions of any of these systems.¹² Three dimensions for evaluating incremental learning methods emerge naturally as a result of adopting this view. The framework and dimensions are used for characterizing the behavior of COBWEB and COBWEB/2, and are themselves a contribution. They encourage experimental validation and comparison of incremental learning methods, something missing in previous accounts.

2.6 A Performance Task for Conceptual Clustering

A performance task that improves with learning is a vital concern of most learning systems. This task motivates choices in system design and is the dimension used to judge the efficacy of learned knowledge. Nevertheless, conceptual clustering has not been traditionally associated with a performance task. With this in mind, an important question is "How do you know the classifications you get are any good?"

¹² Except ID4, whose experimental characterization was guided by this view.

Michalski and Stepp's [MIC83A, MIC83B] original formulation of conceptual clustering stressed it as a way of organizing data in an 'understandable' manner by forming classes that had 'simple' and 'tight-fitting' concepts. The quality of classification schemes was judged by the implementers and domain experts in terms of understandability. The tack of this dissertation is considerably different; classification schemes and concepts are judged by the degree that they promote predictions about the environment.

The idea that conceptual clustering can improve inference is not entirely new. The reason for dropping nodes with too few predictable values in UNIMEM (and CYRUS) is that nothing useful can be predicted from class membership. However, a performance task for UNIMEM is never clearly explicated [LEBO82, LEBO85, LEBO86]. Kolodner asserts that CYRUS and related systems are useful for 'diagnosis' in domains such as psychiatric illness and dispute mediation [KOL85A, KOL85B, KOL83C]. However, she provides no well-defined performance task. There is only anecdotal evidence that these systems improve performance.

A somewhat better delineation of an inference task is given by Cheng and Fu [CHEN85]. They show that in the domain of traditional Chinese medicine, their system builds a classification tree that corresponds well to classification schemes used by experts. Further, after building the classification they attach expert assigned diagnostic conditions to classified instances. They show that of the 118 instances that were used to build the classification tree, 117 are correctly diagnosed using the tree. While their statement and validation of the inference abilities of classification structures is better than that of Lebowitz and Kolodner, there are problems. First, Cheng and Fu only consider prediction of diagnostic condition. If diagnostic condition is deemed equivalent to class membership then this task is identical to the one used to validate learning from examples systems. It is not clear that using conceptual clustering is useful if teacher-defined classes

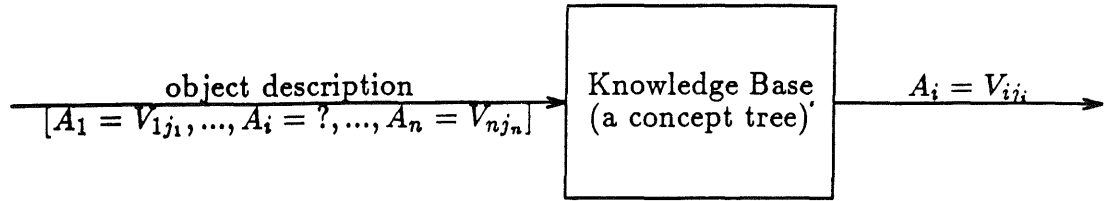


Figure 19

A performance task for conceptual clustering

are already known. Cheng and Fu do little to illuminate this issue. Secondly, they test diagnostic ability on previously observed objects. There is no demonstration that inference ability extends to unseen cases, which is the primary motivation for using inductive techniques.

COBWEB and COBWEB/2 are characterized with respect to the performance task shown in Figure 19. Classification structures are useful in predicting unknown attribute values of an observed object. Success at this task implies that the percentage of correct prediction for arbitrary attributes should be high. This property should hold over previously unobserved instances, as well as those objects used to build the classification scheme. This performance task motivates the selection of a concept quality measure in Chapter 3.

This performance task is a generalization of the one for learning from examples. Like Cheng and Fu, class membership (as defined by some 'teacher') can be treated as simply another attribute for purposes of prediction. Note that this does not imply that the *process* of conceptual clustering is a generalization of learning from examples. Nor does it say anything about whether conceptual clustering will do better, the same, or worse than learning from examples at facilitating prediction of class membership. However, both of these issues will be addressed later.

2.7 Chapter Summary

In this chapter two machine learning tasks, learning from examples and conceptual clustering, were framed as search. Hierarchical conceptual clustering was described as three-tiered search: a search through a space of hierarchies, a search through partition space, and a search for characterizations. This framework was extended to incremental concept induction which was distinguished along two search dimensions, search control and search direction. A strategy of diminished search control (i.e., hill climbing) and bidirectional mobility was abstracted from a number of concept learning systems. This strategy is the basis for COBWEB's search through the space of partitions and hierarchies, processes that are detailed in chapter 4. Additionally, a performance task is suggested for conceptual clustering. The identification of a behavior that improves due to conceptual clustering has not been previously explicated, but is important if conceptual clustering is to remain a vigorous area of machine learning research.

The following chapter introduces important psychological phenomena that motivate a measure of concept quality. While this discussion is important for defining the heuristic measure used by COBWEB, chapter 3 also introduces psychological effects modeled in chapter 7.

Chapter Acknowledgements

While this chapter has been designated as background, some of the discussion is original. Of this, much is owed to the help of others. Characterizing conceptual clustering methods in terms of search was done in conjunction with Pat Langley. This led to a prescriptive view of incremental methods in terms of search and the three dimensions for evaluating these methods. Discussion with Dennis Kibler helped formulate these dimensions and apply them in analyzing COBWEB. Extensive analysis using them was also done with Jeff Schlimmer in the context

of ID4. Independent of this collaboration, Jeff conceived of, designed, and implemented the ID4 family of programs and is responsible for many insights (e.g., the relation between the process of 'approximating backtracking' and dependency directed backtracking). Discussions with Dennis Kibler led to a performance task for conceptual clustering. Talks with Dave Ruby and Rick Granger led to insights into the relation between the performance tasks of conceptual clustering and learning from examples.

CHAPTER 3

Psychological Constraints on Concept Induction

3.1 Chapter Overview

Chapter 2 developed incremental conceptual clustering as a process of search. This framework comes from artificial intelligence and it represents an important conceptual foundation for COBWEB. However, COBWEB classification trees also are the basis of a psychologically consistent model of memory organization. This chapter presents the psychological phenomena that this memory organization models. Moreover, the chapter motivates some high level constraints on the COBWEB system itself, including a concept language that extends the attribute-value representations of the last chapter and a measure of concept quality that is used to guide search.

In section 2, the view of a concept as a logical conjunction of properties is questioned in light of *typicality* (or *prototypicality*) effects that have been observed during human classification. Attempts to account for these effects have led many researchers to adopt a *probabilistic* definition of concepts. This chapter adopts one such definition that augments the attribute-value representation of chapter 2.

Section 3 describes experimental evidence supporting the existence of a *basic* or preferred set of concepts in human hierarchical memory systems. Additionally, explanations of basic level effects are used to develop a measure of concept quality.

In summary, chapter 3 leaves intact the idea that concept induction is a process of search. However, the discussion of chapter 3 opposes commonly held AI views about the nature of the space(s) searched during induction. Specifically, typicality effects impact the representation of individual concepts, while basic-level

effects impact assumptions about the structure of concept hierarchies. Quantitative measures used to account for these phenomena are proposed as heuristics for guiding search in a principled manner.

3.2 Typicality Effects and Probabilistic Concepts

The concept representation discussed in chapter 2 is roughly equivalent to Bruner, Goodnow, and Austin's [BRUN56] definition of a *conjunctive concept*. Concepts of this form fall under what Smith and Medin [SMIT81] have called the *classical view* of concept structure.

One implication of classical concept representations is that all concept members are in some sense 'equal'. To be recognized, each object value must be examined to insure that it is in the value set of the respective concept attribute. Each attribute is checked, regardless of the makeup of the instance being scrutinized; thus the assertion of 'equality' for all concept members. However, work in experimental psychology indicates that human subjects do not treat concept instances equally, but regard certain members as more 'typical' than others.

3.2.1 Experimental Indicators of Typicality Effects

Many studies of human classification are concerned with the ability of subjects to recognize concept instances or subclasses. A *target recognition* task requires a subject to answer questions of the form "Is a *sparrow* a *bird*?" In this example, *bird* is the *target* concept against which *sparrow* is tested for membership. Two variants on this task have been used. The first uses a verbal cue (e.g., the word *sparrow*) to identify the test item. The second uses a picture (e.g., of an actual *sparrow*) to identify the test item. In most all studies, the target is identified verbally.

Regardless of whether the test item is given symbolically (by word) or pictorially, the human subject is required to answer 'yes' if the test item is a member

of the target or 'no' otherwise. The primary variable of interest is the subject's response time. Several studies [RIPS73, ROS75B]¹³ indicate that subjects consistently respond affirmatively more quickly to certain positive instances than to others. For example, subjects more quickly confirm that a sparrow is a bird than they confirm that a chicken is a bird. The relative ranking of positive test items (as a function of response time) is consistent within individuals, as well as across individuals.

The ranking of test items in target recognition tasks is indicative of an apparent bias in human subjects; some test items are more typical representatives of a concept than others. Besides studies of classification time, the belief in a typicality ranking is bolstered by evidence from a variety of other experimental sources. For instance, Rips, Shoben, and Smith [RIPS73] and Rosch [ROS75A] had subjects rank members of a target concept based on the subject's judgements of typicality. The rankings explicitly obtained from subjects are highly correlated with rankings based on response time. In addition, various studies have also found that typical items (judged by the tasks above) tend to be learned first and that when asked to list all members of a particular concept (*exhaustive retrieval*), subjects tended to list items in order of decreasing typicality [ROS76B]. See [MERV81, MERV80, ROSC78, SMIT81] for good reviews.

3.2.2 Implications of Typicality Effects on Concept Structure

Typicality effects suggest that concept members do not interact identically with their respective concept definition. Classical representations do not easily account for these effects and a number of alternative concept representations have been proposed.

An early attempt to discover structural determinants of typicality was made by Rosch et. al. [ROS76B, ROS75B]. They found that class members sharing

¹³ Smith and Medin [SMIT81, p 35] indicate that more than 25 studies of semantic categorization have been conducted and give a partial bibliography.

features with many other members tended to be judged most typical. Further, when a disjoint (or 'contrasting') class was considered, members that shared few features with contrasting class members tended to be judged most typical. Thus, Rosch and Mervis identify two aspects of typicality; typical items share much with other members of the same class and they share little with members of contrasting categories. These two aspects are captured in a *family resemblance* function that assumes a test item, I , a target class, C that contains I , and a set of contrasting classes that are denoted collectively as $\neg C$. Specifically,

$$\text{family-resemblance}(I, C) = f\left(\sum_{C_i \in C} |I \cap C_i|, \sum_{C_j \in \neg C} |I \cap C_j|\right). \quad (3-1)$$

The family resemblance of a class member is a function of the number of properties shared with other members of the same class and the number of properties shared with contrasting class members.¹⁴ Similar properties have been found to relate to human identification of *stereotypes* [McCA80, McCA78].

3.2.3 Probabilistic Concepts

The apparent relation between family resemblance and typicality indicates the importance of attribute value distributions in human classification. Classical concept representations do not capture distributional information. Models of classification based on classical representations have difficulty accounting for typicality effects. A number of concept representations have been developed in response to the limitations of classical representations. A class of these representations have been termed *probabilistic* concept representations by Smith and Medin [SMIT81]. A probabilistic representation associates a probability, weight, or some other confidence number with each attribute value of a concept definition.

¹⁴ $|I \cap C_i|$ is the size of the intersection (i.e., the number of shared properties) between objects I and C_i .

The following discussion assumes a strict probability is maintained. That is, a concept includes attribute values as in classical representations and the distribution of those values across class members as well. For each attribute-value pair, $A_i = V_{ij}$, a conditional probability of the form $P(A_i = V_{ij}|C_k)$ is maintained in the concept representation of class, C_k .¹⁵ $P(A_i = V_{ij}|C_k)$ is referred to as the *category validity* of V_{ij} with respect to C_k . It is a measure of a value's *predictability* with respect to a class. The conventions of chapter 2 (i.e., objects are defined over the same attributes and each object exhibits exactly one value along each attribute) guarantee that for each attribute, A_i , $\sum_{V_{ij}} P(A_i = V_{ij}|C_k) = 1.0$, over V_{ij} in the domain of A_i .

Probabilistic concepts subsume the logical representations of chapter 2. In the case where attribute-value pairs are being used, any probabilistic concept that associates a weight with each pair can be mapped onto a unique (maximally-specific) logical analog. Conversely, for a given logical concept there may be many probabilistic 'equivalents'. A necessary attribute value has a probability of 1.0. Nonnecessary values have probabilities of less than 1.0. Only values with a probability greater than 0.0 are an explicit part of a concept representation.¹⁶

Recognition using probabilistic concepts is not a matter of verifying a conjunction of conditions as with logical representations. It is quite possible that no necessary conditions exist. Somehow, evidence in the guise of individual attribute values must be combined to select one of several possible concepts. A simple recognition scheme is proposed by Smith and Medin [SMIT81]. This procedure sums the probabilities (or weights) of all concept attribute values that are present

¹⁵ This probability is assumed to be exact over all currently known members of C_k . As chapter 4 demonstrates, this probability is only an approximation (hopefully) of the value's distribution over future objects that will be classified by C_k .

¹⁶ Furthermore, nonnecessary, but nonetheless highly probable, values can be interpreted as default values with the associated probability being a measure of confidence in their truth. Default values in the context of probabilistic representations allow for a compact representation of concept knowledge. Further discussion on default values and the relation between probabilistic and logical concept representations is given in chapter 5.

in an object. Specific examples that follow this general recognition procedure include Collins and Loftus [COLL75], Smith, Shoben, and Rips [SMIT74], Hampton [HAMP79], and Hampson [HAMP83]. If the sum passes a specified threshold, the object is assumed to be a member of the concept, otherwise the object is rejected as a concept member.

This model of object recognition offers an explanation of why typical concept members are more quickly recognized than atypical members. The values of typical class members are shared by more objects of the same class. These values have greater probabilities and, as a consequence, summation will tend to reach the threshold more quickly for typical objects. In the case where verbal cues identify a subclass as the test item, the concept corresponding to the cue is assumed to be retrieved by an unspecified process, after which it can be compared with the target (retrieved by the same unspecified process) via the summing procedure. A pictorial representation of the test item is assumed to directly convey the the properties of the test item for inspection by the subject.

At a cursory level the threshold recognition procedure assumes recognition with respect to a concept occurs independently of contrasting concepts. This assumption can be undesirable in cases where concepts must represent mutually disjoint classes. There are at least two ways to extend the summation procedure so that it operates in the presence of contrasting categories. One step is to include a measure in the weight of values that is dependent on the makeup of contrasting concepts. One such measure is *cue validity* [MEDI83, SMIT81]. This is a conditional probability of the form, $P(C_k | A_i = V_{ij})$. It is simply the probability that an arbitrary object is a member of concept C_k , given it exhibits value V_{ij} . Cue validity is a measure of a value's *predictiveness* for a concept. The cue validity of a value with respect to a concept, C_k , is dependent on $\neg C_k$, as well as C_k .

Although a context-dependent measure like cue validity makes classification dependent on contrasting concepts, in general this augmentation cannot enforce a restriction of mutually exclusive classes. A second means of extending the summation procedure is to replace the constant threshold with a variable 'threshold'. An obvious solution is to assume the concept that maximizes the summation for an object is the concept that contains the object.

3.2.4 Alternatives to Probabilistic Concepts

The threshold recognition procedure based on probabilistic concept representations has been called an *independent cue model* of concepts. Recognition is a process of evidence combination that sums separate (or 'independent') pieces of evidence. Medin [MEDI83, MEDI78] asserts that the 'independence assumption' has several important ramifications.

Generally, the real world tends to be segregated into bundles of correlated attribute values [MERV81]. An assumption is that humans are naturally attuned to correlations for categorization. This conjecture has been experimentally supported [MEDI83]. However, independent cue models do not explicitly compute correlations among attribute values. Therefore, the conclusion by some researchers [MEDI83, HANS86] is that independent cue models cannot be reasonable models of human concept structure.

A more specific consequence of using independent cue models is that only linearly separable categories can be recognized. That is, independent cue models assume a weighted and additive combination of independent attribute information. Medin hypothesizes that if independent cue models are reasonable models of human categorization, linearly separable categories should be more easily learned than nonlinearly separable ones. However, experiments [MEDI83] indicate that this is not necessarily the case.

The apparent weaknesses of independent cue models of recognition have led to a number of alternative models. In one way or another, each of these models assume an explicit representation of attribute-value combinations. Recognition profits from knowledge about attribute-value combinations (e.g., red and large), as well as individual values (e.g., red).

3.2.4.1 Exemplar Models

A straightforward way of keeping track of attribute value combinations is to simply remember concept instances. Each instance is a maximally specific conjunction (combination) of values. Summary information can be computed, as necessary, during recognition. These assumptions underlie *exemplar* concept models [SMIT81].

A simple exemplar model is the *proximity* model [REED72]. A concept is an extensional listing of all its known members. A new object is classified as a member of a concept, C, if it most closely matches (i.e., is most similar to or in closest proximity to) an instance of C, versus a member of a contrasting concept. More generally, extensional concept representations are assumed by many techniques of numerical taxonomy. Classification based on a 'best' match with a single known concept member is a principle shared by *single linkage* or *nearest neighbor* techniques of numerical taxonomy [EVER80].

Arguably, keeping and accessing an extensional listing of all known concept instances can become expensive as the number of known objects increases. This is demonstrated by Kibler and Aha [KIBL87] in a comparison between the proximity model with alternative methods. A loosely specified model that limits the number of stored instances is Smith and Medin's *best examples* model [SMIT81]. Smith and Medin assume that a family resemblance function is used to filter out atypical instances. Only instances that are likely to be typical are stored. While reducing the number of instances stored by the proximity model, the best examples model

still allows a significant number of exemplars to be stored. In general, this seems necessary if the structure of many natural categories are to be adequately captured. For example, there may be many 'typical' animals (e.g., a particular dog, a fish, a bird, a frog).

A new instance is matched in parallel against the stored exemplars of each contrasting concept. The first concept for which a specified number of exemplars is found to sufficiently match the instance, according to another threshold, classifies the instance. Presumably, the best examples model would account for typicality effects, but this hypothesis has not been tested. New and typical instances would more closely match the currently stored, typical instances. A criterial number of retrieved exemplars would tend to be reached more quickly for typical instances, rather than for atypical ones.

A final exemplar model is the *context* model by Medin and Schaffer [MED178]. The context model is similar in most respects to the best examples model, including its account for typicality effects. In fact, Smith and Medin developed the best examples model as a simplification of the context model. However, the context model allows the 'importance' or salience of attributes to be specified. Salience can be regarded as the degree that a subject attends to a particular attribute. The impact of variablizing attribute salience is that some attributes can be ignored if salience = 0. For example, if four facial attributes are equally salient, a particular face may be encoded as {Eye-height = high, Eye-separation = wide, Nose-length = long, Mouth-height = low}. However, if subjects do not attend to Mouth-height,¹⁷ the same exemplar may be encoded as {Eye-height = high, Eye-separation = wide, Nose-length = long} [SMIT81, p 152]. Allowing an attribute salience of 0 results in exemplar encodings that are (apparently) abstractions; they may not uniquely match a single object. For example, the encoding {Eye-height = high,

¹⁷ How attribute salience is determined is an issue that falls outside of the definition of the context model.

Eye-separation = wide, Nose-length = long} matches an object with low, medium, or high Mouth-height.

Exemplar models assume that concepts are inherently disjunctive and directly encode attribute-value combinations. Therefore, these models also evade problems like being restricted to representing only linearly-separable categories. The proximity and best examples models remember instance descriptions exactly. These models store attribute-value combinations, but such combinations are limited to complete instance descriptions. The context model is not limited to storing complete instances. A subset of an object's values can be stored. On the surface, this appears to admit abstractions or generalizations over instances. However, in the context model there is no utilization of such a subset as an abstraction; it conveys no information about category-wide regularity whatever. The occurrence of a value subset only implies that one instance with that set of properties occurred. In particular, the salience of an attribute is a measure of importance that is completely independent of category makeup, unlike category and cue validity. The context model uses only instance level, versus 'category level', information for classification. The context model shares this general property with other exemplar models.

3.2.4.2 *Relational Cue Models*

Another class of models are termed *relational cue models* [SMIT81]. Relational cue models generalize both exemplar and independent cue models, albeit in different ways. Like independent cue models, relational cue models allow probabilities (weights, confidence measures) of individual attribute values to be maintained in concept descriptions (e.g., $P(\text{Color} = \text{red} | C_k)$). Moreover, relational cue models also permit joint probabilities for larger configurations of attribute values, e.g., $P(\text{Color} = \text{red} \wedge \text{Size} = \text{large} \wedge \text{Shape} = \text{sphere} | C_k)$. Relational cue models also

generalize exemplar models. Since an instance is a conjunction of values, relational cue models allow the storage of instances (with associated probabilities).

One relational cue model is given by Hayes-Roth and Hayes-Roth [HAYE77]. The *property-set* model supposes that frequencies of all possible value combinations are stored.¹⁸ For example, if objects are described over Size and Shape, a concept for the class, $C_1 = \{\text{large sphere, small cube}\}$, would contain¹⁹

$$\begin{array}{l} \text{large[1], small[1]} \\ \text{sphere[1], cube[1]} \\ \text{large} \wedge \text{sphere[1], small} \wedge \text{cube[1]} \end{array}$$

The decision as to which concept classifies an object is made by identifying the most 'diagnostic' attribute value conjunction among the competing concepts. In particular, from frequencies stored at concepts, the cue validity of each value conjunction can be computed.²⁰ For example, suppose an object, $O = \{\text{small sphere}\}$, is compared with two classes, C_1 (given above) and C_2 . Assume that the only instances of a sphere are in C_1 . Then the cue validity, $P(C_1 | \text{Shape} = \text{sphere}) = \frac{1}{1} = 1.0$. Moreover, suppose that $P(C_2 | \text{Size} = \text{small}) = 0.5$ and $P(C_2 | \text{Size} = \text{small} \wedge \text{Shape} = \text{sphere}) = 0.0$. Then, 'Shape = sphere' is the most diagnostic condition and it indicates that O should be classified as a member of C_1 . There are some rules for deciding 'ties', but in general, recognition using the property-set model is poorly specified and it remains unimplemented.

The property-set model stores frequencies for all possible value combinations for each concept. However, much of this information may be useless for classification. In the example above, $P(C_2 | \text{Size} = \text{small}) = 0.5$ was given, and a simple

¹⁸ While frequencies are explicitly stored, recognition involves computation of probabilities from these frequencies.

¹⁹ It is not clear from the description of Hayes-Roth and Hayes-Roth whether property sets (or value sets) with a frequency of 0 are explicitly stored or implied by their absence.

²⁰ In discussing the independent cue model, the category validity of individual attribute values was assumed. For discussion of relational cue models, it will be convenient to focus on cue validity. However, storing either category or cue validity allows the computation of the other by Bayes' rule.

computation will verify that $P(C_1 | \text{Size} = \text{small})$ must therefore equal 0.5 as well. Knowing that an object is small does not alone aid classification, since the cue validity of this property is the same for each concept. A reasonable storage strategy would throw out value conjunctions that do not aid classification. This strategy is employed in ACT by Anderson and Kline [ANDE79]. ACT assumes structured object descriptions and associates a real-valued weight with predicate conjunctions. While many conjunctions are stored, only those whose weight surpasses a variable threshold are used to generate a prediction. A more principled approach to filtering out minimally diagnostic conditions is used by STAGGER, a system by Schlimmer and Granger [SCH86c]. STAGGER weights attribute value combinations (i.e., conjunctions, disjunctions, negations) using statistical analogs of logical sufficiency and logical necessity that were originally used in the PROSPECTOR system [DUDA79]. These measures interact so that only useful diagnostic combinations tend to be retained.

Recognition in relational cue models is based on weights of individual attribute values, as well as combinations of values. This allows computation of nonlinearly separable categories and makes recognition a function of correlated values. However, computing joint probabilities can be expensive. It may be necessary to forego the luxury of computing all joint probabilities as in the property-set model and only keep certain value combinations. The heuristics that determine what conditions to remember will differ from system to system.

3.2.4.3 Summary

In general, the representational power of exemplar-based and relational cue models are equivalent. Stored exemplars can be used, as needed, to compute any of the information used in relational cue models. To a limited extent, joint probabilities are computed from exemplars by a number of current clustering systems [HANS86].

Exemplar and relational cue models both address a purported weakness of independent cue models. Independent cue models are limited in that they do not allow correlations between attribute values to be represented. However, the approach of this dissertation is that this motivates a consideration of *concept organizations*, not new models of individual concepts. In particular, concepts of the independent cue variety capture the same information as exemplar and relational cue models, provided they are organized into concept hierarchies.

3.2.5 Independent Cue Models and Concept Hierarchies

Smith and Medin's complaints regarding independent cue models (henceforth, probabilistic concepts) dissipate when concepts are not considered singly, but within the context of larger constructs like concept trees. Trees can be used to compute the information content of relational attribute value information. In terms of information content, tree-organized probabilistic concepts are equivalent to relational cue models.

3.2.5.1 Equivalence of Relational Cue Models and Probabilistic Concept Trees

Consider a situation with two contrasting concepts, C_1 and C_2 . Assume that the diagnostic information retained by a relational cue model includes:

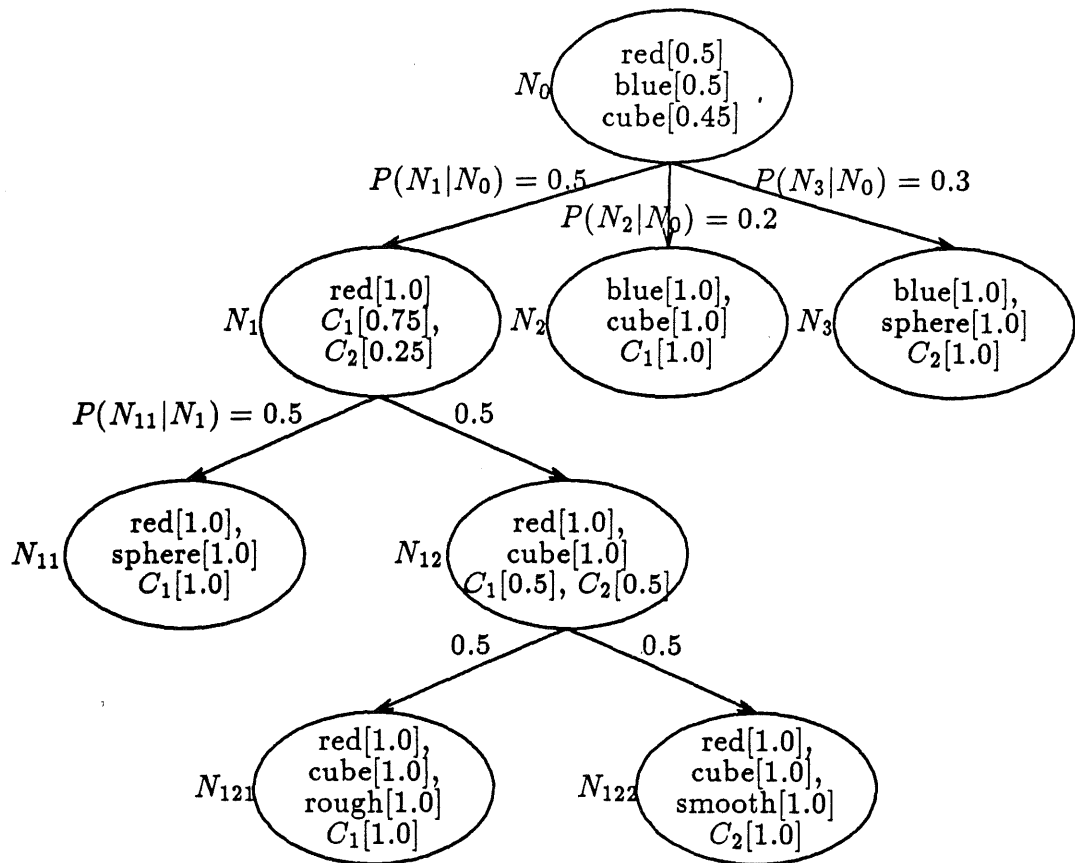


Figure 20

Encoding joint probabilities in tree form

$$\begin{aligned}
 P(C_1|\text{red}) &= 0.75 \\
 P(C_2|\text{red}) &= 0.25 \\
 P(C_1|\text{red} \wedge \text{cube}) &= 0.5 \\
 P(C_2|\text{red} \wedge \text{cube}) &= 0.5 \\
 P(C_1|\text{red} \wedge \text{sphere}) &= 1.0 \\
 P(C_1|\text{red} \wedge \text{cube} \wedge \text{rough}) &= 1.0 \\
 P(C_2|\text{red} \wedge \text{cube} \wedge \text{smooth}) &= 1.0 \\
 P(C_1|\text{blue} \wedge \text{cube}) &= 1.0 \\
 P(C_2|\text{blue} \wedge \text{sphere}) &= 1.0
 \end{aligned}$$

This information can be encoded as shown in in Figure 20. Associated with each node is a probabilistic concept that gives the node-conditioned category validity of each attribute. However, for convenience most of the nodes of

Figure 20 only show values with probability 1.0 (e.g., $P(\text{red}|N_1) = 1.0$). Class designations and their associated probabilities (e.g., $P(C_1|N_1) = 0.75$) are shown as well. In addition, some supplementary information is given, like the base rate probabilities of values (e.g., $P(\text{red}|N_0) = 0.5$) and the probability of each node (e.g., $P(N_1|N_0) = 0.5$).

Each node of Figure 20 represents an object class. This type of tree will be called a *probabilistic concept tree*. It differs from a decision or discrimination tree [FEIG84] in that probabilistic (and not logical) concepts label nodes (and not arcs) of the tree. Classification is performed by using a partial matching function (e.g., a summing procedure) to find the class that best matches an object. This process is described in more detail later.

Concepts at probabilistic concept tree nodes capture the diagnostic conditions of the example relational cue model. For example, N_1 , N_2 , and N_3 cover only and all cases of 'red', 'blue \wedge cube', and 'blue \wedge sphere', respectively. More generally, from individual value probabilities at each node (and node probabilities), information about attribute value combinations can be computed. For example, probabilities of the form $P(C_i|\text{condition})$ can be recovered from independent attribute value information at different nodes. Specifically,

$$P(C_i|\text{condition}) = \sum_j P(C_i \wedge N_j|\text{condition}), \quad (3-2)$$

where N_j is a first level node. Applying Bayes' rule, each term of this sum can be rewritten as

$$P(C_i \wedge N_j|\text{condition}) =$$

$$\frac{P(\text{condition}|C_i \wedge N_j)P(C_i \wedge N_j)}{P(\text{condition})} = \frac{P(\text{condition} \wedge C_i|N_j)P(N_j)}{P(\text{condition})}. \quad (3-3)$$

For some value conjunctions, the tree has been structured so as to make the computation of 3-2 relatively inexpensive. Other combinations are not so cleanly captured. Moreover, the difficulty of computing $P(\text{condition} \wedge C_i | N_j)$ for each N_j can vary. The following examples reflect this diversity.

Example 1

Consider the computation of $P(C_1 | \text{blue} \wedge \text{cube})$:

$$\begin{aligned} P(C_1 | \text{blue} \wedge \text{cube}) &= P(C_1 \wedge N_1 | \text{blue} \wedge \text{cube}) + \\ &P(C_1 \wedge N_2 | \text{blue} \wedge \text{cube}) + \\ &P(C_1 \wedge N_3 | \text{blue} \wedge \text{cube}) \end{aligned}$$

In this case, $P(C_1 \wedge N_1 | \text{blue} \wedge \text{cube}) = P(C_1 \wedge N_3 | \text{blue} \wedge \text{cube}) = 0.0$. Evaluating $P(C_1 \wedge N_2 | \text{blue} \wedge \text{cube})$ using 3-3 gives

$$P(C_1 \wedge N_2 | \text{blue} \wedge \text{cube}) = \frac{P(\text{blue} \wedge \text{cube} \wedge C_1 | N_2) P(N_2)}{P(\text{blue} \wedge \text{cube})}$$

Because N_2 contains all and only 'blue \wedge cube' objects, $P(N_2) = P(\text{blue} \wedge \text{cube}) = 0.2$ and $P(\text{blue} \wedge \text{cube} | C_1 \wedge N_2) = 1.0$. So

$$P(C_1 | \text{blue} \wedge \text{cube}) = \frac{1.0 \times 0.2}{0.2} = 1.0$$

The tree of Figure 20 is organized so that computation of $P(C_1 | \text{blue} \wedge \text{cube})$ is easy. Specifically, the $P(\text{blue} \wedge \text{cube} | N_2) = 1.0$ implies $P(\text{blue} \wedge \text{cube} \wedge C_1 | N_2) = 1.0 \times P(C_1 | N_2)$. Similar circumstances make the computation of $P(C_1 | \text{red}) = 0.67$ and $P(C_2 | \text{blue} \wedge \text{sphere}) = 1.0$ equally easy. Even nodes at deeper levels of the tree facilitate a straightforward computation of diagnosticity. For example, knowing that $P(N_{122}) = P(N_{122} | N_{12}) P(N_{12} | N_1) P(N_1 | N_0)$ enables a simple verification that $P(C_2 | \text{red} \wedge \text{cube} \wedge \text{smooth}) = 1.0$.

Example 2

By isolating joint probabilities at nodes, trees ease the task of computing certain relational information. However, 3-2 indicates that to compute some joint probabilities requires an examination of multiple paths. For example, $P(C_1|\text{cube})$ can be computed, but it is more costly to do so than the previous examples. To compute $P(C_1|\text{cube})$ requires finding a path to all nodes, N_i , such that $P(\text{cube}|N_i) = 1.0$, i.e., N_{12} and N_2 . From information at these nodes and accumulated along the paths to these nodes, $P(C_1|\text{cube})$ can be computed. In particular,

$$P(C_1|\text{cube}) = P(C_1 \wedge N_1|\text{cube}) + P(C_2 \wedge N_2|\text{cube}) + P(C_1 \wedge N_3|\text{cube}).$$

$P(C_1 \wedge N_3|\text{cube})$ simply equals 0 and further exploration of this node can be discontinued. Since $P(\text{cube}|N_2) = 1.0$,

$$P(C_1 \wedge N_2|\text{cube}) = \frac{P(\text{cube} \wedge C_1|N_2)P(N_2)}{P(\text{cube})} = \frac{1.0 \times 0.2}{P(\text{cube})}$$

However, since $P(\text{cube}|N_1) \leq 1.0$,

$$P(C_1 \wedge N_1|\text{cube}) = \frac{P(\text{cube} \wedge C_1|N_1)P(N_1)}{P(\text{cube})} = \frac{P(\text{cube} \wedge C_1|N_1) \times 0.5}{P(\text{cube})}$$

where $P(\text{cube} \wedge C_1|N_1)$ must be evaluated in terms of N_1 's children.

$$\begin{aligned} P(\text{cube} \wedge C_1|N_1) &= P(N_{11}|N_1)P(\text{cube} \wedge C_1|N_{11}) + P(N_{12}|N_1)P(\text{cube} \wedge C_1|N_{12}) \\ &= (0.5 \times 0.0) + (0.5 \times 0.5) \\ &= 0.25 \end{aligned}$$

$P(\text{cube})$ is given at the root. However, in general $P(\text{condition})$ can be computed by summing the probabilities of nodes for which 'condition' is true with probability 1.0.²¹ Thus, $P(\text{cube}) = P(N_1|N_0)P(N_{12}|N_1) + P(N_2|N_0) = 0.45$. All told,

$$P(C_1|\text{cube}) = \frac{0.25 \times 0.5}{0.45} + \frac{1.0 \times 0.2}{0.45} = 0.72$$

In general, any joint probability can be computed from tree-structured probabilistic concepts. In terms of information content, tree-structured probabilistic concepts are equivalent to relational cue or exemplar models.²² In the best case, a single tree node isolates a desired conjunction of values; the probability of the value conjunction is the probability of the node. In the worst case, an extensive search of the tree may be necessary to compute joint probabilities from subordinate nodes, including leaves representing instances. Ideally, the tree should capture diagnostic correlations that occur most frequently, thus minimizing average computational requirements.

3.2.5.2 Concept Tree Nodes and Conditional Independence

The computations above rely on nodes where $P(\text{condition}|N_j) = 1.0$. These nodes provide a simple way of calculating $P(\text{condition} \wedge C_i|N_j) = 1.0 \times P(C_i|N_j)$, i.e., 'condition' is true in the context of N_j regardless of the presence of C_i or not. Therefore the joint probability of 'condition' and C_i conditioned on N_j is simply the probability of C_i conditioned on N_j . The case where $P(\text{condition}|N_j) = 1.0$ insures that 'condition' is *conditionally independent* of all other conditions, including the truth of C_i . However, the conditional probability of a value conjunction need not be 1.0 to insure its conditional independence from other attributes. In fact, many

²¹ Using probabilistic concepts at nodes, the probabilities of a conjunction will not be stored at the root.

²² Analogous and more formal arguments for the equivalence of networked *perceptrons* and other forms of computation are given in [NILS65, ROSE58].

useful categories have no necessary values, but nonetheless have typical values that approximate independence from other class defining values.

Pearl [PEAR85] points out that classification tree nodes, N_j , act as ‘auxiliary’ attributes that may induce conditional independence over some observed attribute(s). Specifically, for a node, N_j , and all values of attributes, A_{i_1} through A_{i_m} ,

$$P(A_{i_1} = V_{i_1 j_{i_1}}, \dots, A_{i_m} = V_{i_m j_{i_m}} | N_j) = \prod_{l=1}^m P(A_{i_l} = V_{i_l j_{i_l}} | N_j).$$

In general, computing joint probabilities benefits from nodes that induce conditional independence among attributes and class designation.

3.2.5.3 Prediction Using Probabilistic Concept Trees

Probabilistic concept trees encode attribute correlations at nodes. In particular, discussion has focussed on category diagnostic information. A tree that captures ‘appropriate’ correlations can be used for effective diagnosis or prediction of category membership. Recalling discussion from chapter 2, prediction is a byproduct of classification with respect to the tree.

Using a tree, an object is initially classified with respect to the children of the root. Using logical (‘classical’) concepts, chapter 2 assumed that this involved determining an exact match between the object and arc-labeling concepts. However, using probabilistic concepts, classification is made via a partial match using a summing procedure. If the object is to be classified with respect to only one child, the node that maximizes the summation is regarded as classifying the object. Classification proceeds recursively until a leaf or an appropriate intermediate node is reached. At this point a prediction of category membership is made.

For the tree of Figure 20, classification results in a prediction of either C_1 or C_2 . Node N_2 covers objects for which the conjunction ‘blue \wedge cube’ is true. Presumably, a new blue, cubical object stands a good chance of being classified

with respect to N_2 , although this depends on the relation of all the object's values with each node. Having classified an object with respect to N_2 , the best guess as to its class membership is C_1 . For similar reasons, it is probable that a red, rough, cube will be classified with respect to N_{121} . However, unlike the previous example, the diagnosticity of attribute values that indicate N_{121} are spread out over a number of nodes. That is, $P(N_1|\text{red}, N_0) = 1.0$, $P(N_{12}|\text{cube}, N_1, N_0) = 1.0$, $P(N_{121}|\text{rough}, N_{12}, N_1, N_0) = 1.0$.

The procedure of accumulating conditions over a path of nodes is also followed by decision trees. Recall (from chapter 2) that arc-labeling conditions of a decision tree perfectly segregate objects at each node. An exact match is required if an object is to traverse an arc. However, this representation and recognition procedure can be easily simulated using probabilistic concepts. For example, assume that the probabilities of 'red', 'cube', and 'rough' at N_1 , N_{12} , and N_{121} , respectively, are replaced by sufficiently high weights so that they dominate any possible summation. As with a decision tree, a single value will determine categorization – other value probabilities at nodes will not significantly effect the categorization process. Decision trees are a special case of probabilistic concept trees, in which necessary and sufficient values are introduced at each node.

In general, trees encode diagnostic information that can be retraced during classification. With perfect knowledge about the world, a tree can be constructed so that classification and prediction are deductive. Trivially, a decision tree (which can be simulated by a probabilistic concept tree) over all objects of a particular domain perfectly distinguishes the class membership of every domain object. However, the application of conceptual knowledge is rarely limited to deductive tasks. Frequently, concepts and concept trees are used for induction tasks, for example, predicting the class membership of previously unseen objects.

For inductive tasks, probabilistic concept trees may pose limitations. Some correlations are distributed along a single path, but generally, multiple paths must be explored to recover arbitrary correlations. However, by convention, classification using trees proceeds along a single path of best matching nodes. Under these circumstances important diagnostic information may be excluded from the prediction process. In principle trees encode all relational information, but in practice, all such information may not be recovered during classification and prediction. One solution allows multiple paths to be explored during classification [QUI87B]. If object *incorporation* is similar to *classification*, multiple path exploration effectively leads to more general hierarchical structures. Apparently, this is a major motivation behind the clumping techniques of Lebowitz [LEBO82] and Kolodner [KOL83A]:

3.2.5.4 Probabilistic Concept Trees and Nonlinearly Separable Classes

A specific result of not representing attribute value correlations is that probabilistic concepts are limited to recognizing linearly separable classes. Medin [MED183] assumes that if independent cue models are good models of human concept structure, linearly separable categories should be easier to learn than nonlinearly separable sets. An investigation of this question required that subjects learn (from examples) one of the two category pairs of Table 4. Objects were characterized in terms of four binary attributes, A_1 through A_4 . Learning the linearly separable set resulted in more recognition errors and was judged by subjects more difficult to learn. Thus, learning linearly separable sets is not *necessarily* easier than learning nonlinearly separable sets.

Medin's experimental results are only damning of probabilistic concepts that are considered singly. Probabilistic concept *trees* are not constrained to recognizing linearly separable sets. In fact, Medin's experimental results are easily accounted for by assuming that probabilistic concept trees are learned, rather than solitary concepts.

 LINEARLY SEPARABLE CATEGORIES

		CATEGORY C_1				CATEGORY C_2				
		A_1	A_2	A_3	A_4	A_1	A_2	A_3	A_4	
Object	1)	1	1	1	0	5)	1	0	1	0
	2)	1	0	1	1	6)	0	1	1	0
	3)	1	1	0	1	7)	0	0	0	1
	4)	0	1	1	1	8)	1	1	0	0

NON-LINEARLY SEPARABLE CATEGORIES

		CATEGORY C_1				CATEGORY C_2				
		A_1	A_2	A_3	A_4	A_1	A_2	A_3	A_4	
	9)	1	0	0	0	13)	0	0	0	1
	10)	1	0	1	0	14)	0	1	0	0
	11)	1	1	1	1	15)	1	0	1	1
	12)	0	1	1	1	16)	0	0	0	0

Table 4

 Linearly separable and non-linearly separable classes

Consider the concept trees of Figure 21, which discriminate the category pairs of Medin's experiments. An independent cue model insists that each node divides the total object set into linearly separable categories. However, this division need not correspond to the sets that were taught, C_1 and C_2 . Rather, members of these classes may reside in distinct portions of the classification tree. Probabilistic concepts at internal nodes are abbreviated by a general pattern (e.g., 101X) that lists values occurring with probability 1.0 and placing an 'X' for attributes in which

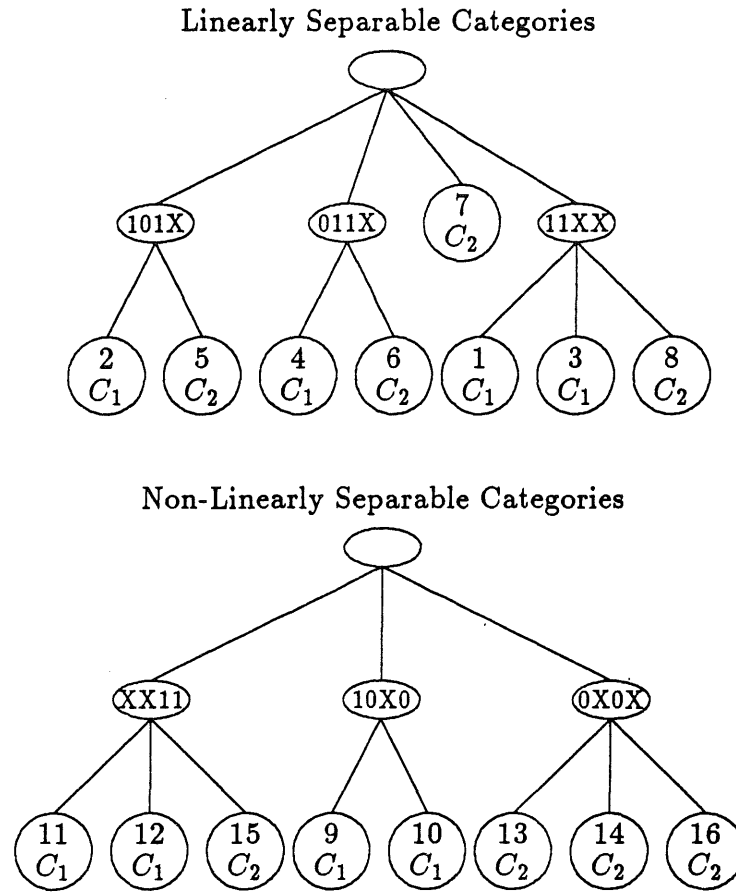


Figure 21

Concept trees over non-linearly and linearly separable sets

no value occurs with probability 1.0. The tree reveals a peculiarity of the linearly separable set; objects in different categories, C_1 and C_2 , share many properties. It is reasonable to expect that they might be placed under the same tree node. Object 7 of the linearly separable set is quite unlike any other members of its category, C_2 . On the other hand, fairly specific patterns can be found that perfectly discriminate many members of the opposing categories for the nonlinearly separable set.

Discrimination of C_1 and C_2 stems from an ability to descend the tree to arbitrary depth before making a decision as to class membership. This descent may be all the way to the leaves of the tree where information on individual objects

are kept. However, intermediate nodes may provide sufficient information (e.g., $P(C_i|N_j) = 1.0$) to warrant a prediction. This occurs with two intermediate nodes from the tree over the nonlinearly separable data. In these cases classification need only proceed to a depth of 1 for a correct decision to be made with certainty. Medin's finding that the nonlinearly separable sets have an associated fewer number of recognition errors might result as a function of the average depth required to distinguish whether an object is from class C_1 or C_2 (depth 1.37 for non-linearly separable sets vs. 1.87 for the linearly separable sets in the sample trees).

3.2.5.5 *Concept Trees, Exemplar Models, and Relational Cue Models*

Concept trees that adopt an independent cue model of individual concept (i.e., node) structure provide the same information as exemplar or relational cue models. Intuitively, a concept tree can include all observed objects at its leaves. Arbitrary joint distributions can be computed from these instances. However, a concept tree captures certain attribute value correlations at nodes. This makes the computation of some relational information easy. It also imposes an classification scheme over instances. In this light, concept trees can be viewed as a more tightly constrained specification of either exemplar or relational cue models, rather than an alternative to them. An appropriately structured tree can efficiently implement the classification procedures of these models.

An observation that is important for understanding the equivalence of concept trees to other concept models is that two types of categories result from organizing preclassified instances into a tree. First, categories may be defined by some external source or 'teacher'. These categories are not restricted to being linearly separable. The only constraint is that constituent objects must be expressible in a given object language (e.g., attribute-value pairs). Ongoing discussion has denoted these classes as C_i . Second, there are categories that correspond to classification tree nodes, N_j . Concepts at tree nodes fit the independent cue model of concept structure. Their

respective categories are linearly separable. Throughout the dissertation, there will be some need to explicitly address the computational and psychological differences associated with each type of concept. For example, in cases where objects come preclassified, concept tree organization can reflect a desire to maximize correct prediction for teacher-defined classes (C_i). However, this dissertation is concerned with concept formation where objects are not preclassified, but arise naturally. In concept trees like those discussed here, naturally occurring concepts correspond to nodes (N_j).²³ Criteria for tree structuring will differ from the case where preclassification exists.

From a psychological standpoint, studies involving artificial domains require subjects to first learn relevant categories by examples. Typicality findings apply to the externally defined classes. Other studies use natural domains (e.g., animals). These categories correspond roughly to single nodes. Results obtained in future chapters will apply most strongly to natural or 'node concepts', but, to a limited extent, will also address computational and psychological issues relating to externally defined categories.

3.3 Basic Level Effects and Concept Hierarchies

Within hierarchical classification schemes there appears to be a *basic* level that human subjects tend to prefer. For example, in a hierarchy containing {animal, vertebrate, mammal, dog, collie}, the behavior of most human subjects indicates that 'dog' lies at the basic level. The identification of preferred concepts in humans must impact any model of human hierarchical classification. It also provides a basis for developing principled criteria for evaluating concept quality in the contexts of concept learning and recognition.

²³ In this light, an informative follow-up to Medin's study [MED183] would focus on the types of concepts naturally formed by subjects in a concept formation setting. An initial hypothesis is that discovered concepts would be linearly separable.

3.3.1 Experimental Indicators of Basic Level Effects

Many experimental studies of basic level effects are similar in form those focusing on typicality. For example, Rosch [Ros76A] used a target recognition task to show that subjects are quicker to confirm that a test item is a member of a basic level concept versus a superordinate or subordinate concept. For example, if a collie (i.e., the test item) is shown to a subject and the subject is asked "Is this a dog?", the subject will more quickly confirm that it is a dog than confirmation will be given for similar queries on whether it is a collie (subordinate) or animal (superordinate).

The nodes of a classification tree that are 'preferred' in the target recognition task tend to be the same nodes preferred in other tasks as well. In a *forced naming task* [Ros76A, JOL184], a subject is shown a picture of a particular entity (a collie again) and asked to respond with its identity (rather than confirming the correctness of a given identity). When a picture of a collie is shown, most subjects will respond that it is a dog, rather than a collie, mammal, or animal.

3.3.2 Implications of Basic Level Effects on Hierarchical Classification

In modeling basic level effects, this dissertation will presume that concepts correspond to nodes of a concept tree. Hierarchical classification typically assumes that classification moves strictly from the root to a leaf. This assumption does not easily account for basic level effects. Assumptions about classification hierarchy structure and/or processing must be reconsidered.

Briefly, results of target recognition and forced naming studies indicate that recognition probably does not proceed in a strict top-down fashion; if so, objects would be recognized more quickly with respect to superordinate categories. Rather, there appears to be an intermediate 'entry point' into the tree that corresponds to the basic level.

It is possible that entry into a hierarchy is 'hard coded' to be the basic level, but closer examination indicates that this is unlikely. Jolicoeur, Gluck, and Kosslyn [JOL184] found that the exact level of entry can vary as a function of typicality. In particular, atypical members of a category are sometimes identified first with respect to a subordinate category. For example, in a forced naming task, a robin (typical bird) will be identified as a bird, while a chicken (atypical) will be identified as a chicken. This suggests that basic level entry is not 'hard coded' but generally emerges as the result of an evidence combination process. This is consistent with some types of hierarchical recognition (e.g., UNIMEM [LEBO82]) discussed in chapter 2, but suggests that discrimination arcs be allowed to skip levels of a tree.

Presumably, once the hierarchy has been entered (generally at the basic level), recognition can proceed to superordinate (upper) or to subordinate (lower) nodes. Superordinate nodes can be reached by climbing IS-A links from the entry point.²⁴ Subordinate nodes can be attained by recursively applying the evidence combination procedure at the entry point. The consistency of this hypothesis is verified by experimental evidence by Jolicoeur, Gluck, and Kosslyn [JOL184], as well as Rosch, et. al. [ROS76A]. Both studied the time differential between classifying (target recognition) an object with respect to its basic level concept and either a corresponding subordinate or superordinate concept. Results showed that significantly more time was required to recognize an object with respect to a subordinate concept than was required for superordinate recognition. This is consistent with, but does not necessarily validate, the view that downward movement involves a relatively expensive evidence combination procedure, while upward movement is a deterministic process guided by IS-A links.

²⁴ An IS-A link connects a node to its parent. IS-A links allow upward movement in a hierarchy. In a tree there is exactly one IS-A link emanating from a node.

3.3.3 Measures for Predicting the Basic Level

A hope of identifying preferred concepts in humans is that measures of concept quality can be developed. In fact, a number of measures have been proposed to explain basic level phenomena. These measures are functions of the internal structure of categories and tend to reward intra-category similarity and inter-category dissimilarity.

An early proposal for predicting the basic level of a hierarchy was given by Rosch [Ros76A, Rosc78]. Rosch postulated that a basic level category in a tree maximizes a function of the cue validity (i.e., $P(N_k|A_i = V_{ij})$) of individual attribute values among its ancestors and descendants. Precisely, Rosch [Rosc78] defines the *total cue validity* of a category as the sum of cue validities for each attribute value which is shared by *all* or *most* category members. The qualification of calculating total cue validity from only attribute values that are shared by all or most category members is left fuzzy by Rosch. Nonetheless, the total cue validity of a category N_k can be approximately formalized as

$$\sum_i \sum_j P(N_k|A_i = V_{ij}) \text{ for all } V_{ij} \text{ s.t. } P(A_i = V_{ij}|N_k) \approx 1.$$

Rosch postulates that basic level categories maximize total cue validity.²⁵ Although Rosch's formulation is not mathematically formal, nor is it rigorously evaluated with respect to experimental findings, it does represent a tradeoff between intra-category similarity and inter-category dissimilarity. Calculating total cue validity using attribute values with high $P(A_i = V_{ij}|N_k)$ (i.e., category validity) tends to reward intra-category similarity or the predictability of individual values,

²⁵ It is important to note that some analyses [MED183] of Rosch's measure do not reflect Rosch's implication that only values with high category utility be used in predicting basic level categories. While this omission seems to shortchange Rosch's formulation, the conclusion of these analyses – a measure that is only dependent on cue validity *cannot* be useful *per se* – is absolutely correct. A measure of class quality, particularly one that predicts basic level preferences, must be dependent on both cue and category validity, or some analogous tradeoff.

while high $P(N_k|A_i = V_{ij})$ (i.e., cue validity) favors inter-category dissimilarity or the predictiveness of values. Maximizing total cue validity also offers an intuitive explanation of basic level effects. Superordinate categories typically will have fewer attribute values common to many instances (fewer values with high $P(A_i = V_{ij}|N_k)$). Subordinate categories will tend to have values that are common to contrasting categories (more values with low $P(N_k|A_i = V_{ij})$). Although maximizing total cue validity has some desirable explanatory properties, it is a relatively ad hoc and underspecified function.

A measure called *collocation* has been proposed by Jones [JONE83] to predict basic level categories. Collocation is a mathematically sounder formulation of Rosch's intuitive notions. The collocation of an attribute value, V_{ij} , with respect to a category, N_k , is the product of the cue and category validities of the value, or

$$\text{collocation}(V_{ij}, N_k) = P(N_k|A_i = V_{ij})P(A_i = V_{ij}|N_k).$$

Jones suggests that a basic level node (e.g., 'bird') has the most collocation maximizing values among all possibilities (e.g., 'animal', 'bird', 'robin'). That is, each category receives a score that is the number of values for which collocation is maximized at that category (as opposed to superordinate or subordinate categories). The category with the highest score is predicted to be the basic category. For example, hypothetical collocation scores for 'Mode-of-transport = flies' are given below.

node	collocation
'animal'	$P(\text{animal} \text{flies}) \times P(\text{flies} \text{animal}) = 1.0 \times 0.25 = 0.25$
'bird'	$P(\text{bird} \text{flies}) \times P(\text{flies} \text{bird}) = 0.95 \times 0.89 = 0.85$
'robin'	$P(\text{robin} \text{flies}) \times P(\text{flies} \text{robin}) = 0.09 \times 1.0 = 0.09$

The collocation for the value, 'flies', is maximized at 'birds'. If 'has-feathers' and 'four-chambered-heart' are maximized at 'birds' as well, then 'birds' would receive a score of 3. If 'animals' only maximized the collocation of 'animate' (score = 1) and 'red' is the only value maximized at 'robin' (score = 1), then Jones' method predicts that 'birds' is the basic level node.

Jones does not empirically validate his measure against experimental evidence, but does make a normative argument in favor of the measure. Intuitively, collocation represents a tension between intra-category similarity (i.e., through $P(A_i = V_{ij}|N_k)$) and inter-category dissimilarity (i.e., through $P(N_k|A_i = V_{ij})$) along the same lines as Rosch. Further, Jones formally shows that collocation arises as a special case of the *index of mean square contingency*, a measure of association in certain numerical taxonomy applications.

A final measure, *category utility*, for predicting basic level concepts has been proposed by Gluck and Corter [GLUC85]. While Gluck and Corter take a different tack in developing category utility, the measure can be presented in terms of Jones' collocation measure. Specifically, for a concept, N_k ,

$$\sum_i \sum_j P(A_i = V_{ij})P(N_k|A_i = V_{ij})P(A_i = V_{ij}|N_k), \quad (3-4)$$

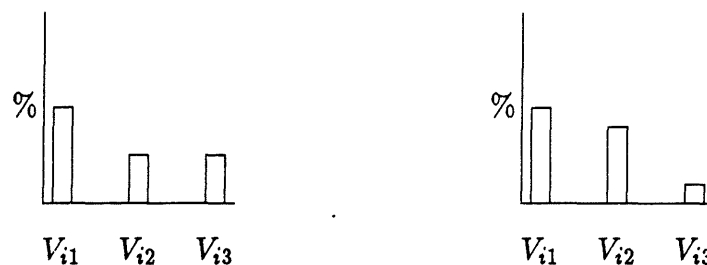
represents a 'weighted' collocation measure that has been summed over attributes (i), and values (j). The base rate probability, $P(A_i = V_{ij})$, weights the importance of individual values, in essence saying that it is more important to increase the class-conditioned predictability and predictiveness of frequently occurring values than infrequently occurring values.

Function 3-4 balances traditional concerns of intra- and inter- class similarity. However, 3-4 can also be regarded as rewarding the inference potential of object class partitions. More precisely, note that for any i , j , and k , that $P(A_i = V_{ij})$

$P(N_k|A_i = V_{ij}) = P(N_k)P(A_i = V_{ij}|N_k)$ (Bayes rule), so by substitution function 3-4 equals

$$P(N_k) \sum_i \sum_j P(A_i = V_{ij}|N_k)^2. \quad (3-5)$$

$\sum_i \sum_j P(A_i = V_{ij}|N_k)^2$ is the *expected* number of attribute values that can be correctly guessed for an arbitrary member of class, N_k . This expectation assumes a guessing strategy that is *probability matching*, meaning that an attribute value is guessed with a probability equal to its probability of occurring. Thus, it assumes that a value is guessed with probability $P(A_i = V_{ij}|N_k)$ and that this guess is correct with the same probability. A probability matching strategy can be contrasted with a probability maximizing strategy, which assumes that the most frequently occurring value is always guessed.²⁶ While this strategy may seem superior at a cursory level, it is *not superior in terms of heuristically ordering object classes*. In particular, a probability maximizing strategy is not sensitive to the *distribution* of all attribute values, but only the most frequent one. For example, assume two distributions of values (V_{ij}) for an attribute, A_i :



A probability maximizing strategy will perform equally well with each distribution. A probability matching strategy will do better with the second distribution than with the first. A heuristic measure motivated by a probability matching strategy

²⁶ A number of studies indicate that human subjects use a probability matching strategy to make predictions. See Bruner, Goodnow, and Austin [BRUN56, pp 182-195] for a discussion of psychological motivations behind a selection of guessing strategies.

will favor a category with distributions like the second. This property is important when constructing classification *trees*. The second distribution provides 'better' ways of decomposing a class to facilitate inference at lower tree levels, using either a probability matching or probability maximizing (!) strategy.²⁷

Gluck and Corter define category utility to be a measure of the *increase* in the expected number of attribute values that can be correctly guessed ($P(N_k) \sum_i \sum_j P(A_i = V_{ij}|N_k)^2$) given knowledge of a category, N_k , over the expected number of correct guesses with no such knowledge. Formally,

$$P(N_k) \sum_i \sum_j P(A_i = V_{ij}|N_k)^2 - P(N_k) \sum_i \sum_j P(A_i = V_{ij})^2. \quad (3-6)$$

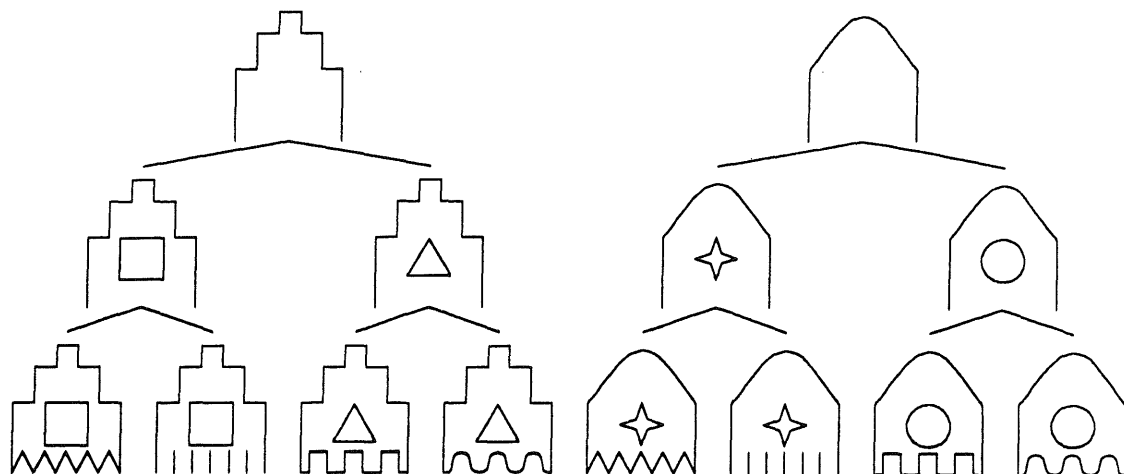
The quantity, $\sum_i \sum_j P(A_i = V_{ij})^2$, is the expected number of correctly inferrable properties with no knowledge of an object partition. $P(A_i = V_{ij})$ is the probability of an attribute value not conditioned on class membership (i.e., the base rate probability).

Function 3-6 can be used in the same manner as collocation or total cue validity to distinguish the basic level node in an ancestral line (e.g., 'animal', 'bird', 'robin'). However, in later discussion, category utility will be used to predict which tree *level* (object set partition) of several competing tree levels is the basic level. This is done by averaging over all categories of a partition. That is,

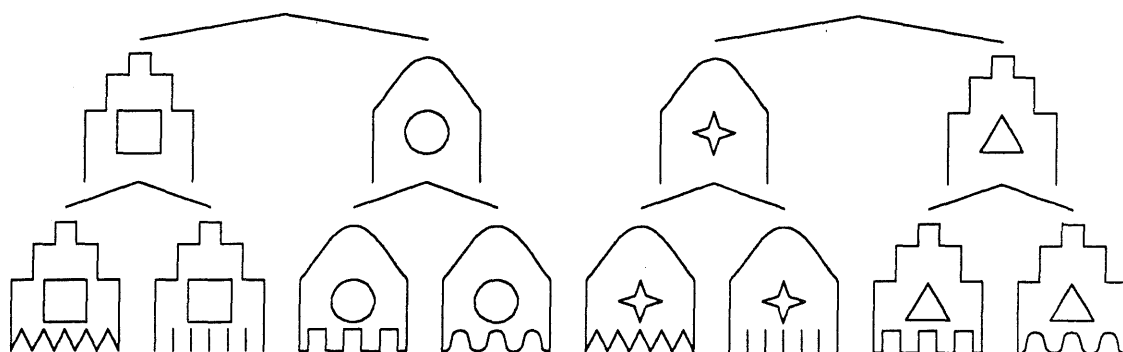
$$CU(\{N_1, \dots, N_n\}) = \frac{[\sum_{k=1}^n P(N_k) \sum_i \sum_j P(A_i = V_{ij}|N_k)^2] - \sum_i \sum_j P(A_i = V_{ij})^2}{n} \quad (3-7)$$

²⁷ Assuming a probability *matching* strategy to build classification trees also facilitates correct prediction using a probability *maximizing* strategy. Experimental evidence in chapter 5 bears this out.

Hierarchy #1



Hierarchy #2



Hierarchy #3

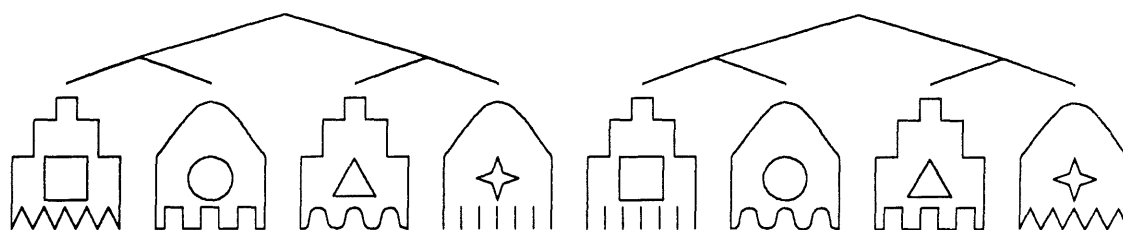


Figure 22

Hierarchies of objects defined by outer, inner, and bottom shape

The denominator, n , is the number of categories in a partition, and averaging over categories allows comparison of different size partitions.

Gluck and Corter show that category utility consistently predicts the basic level (as behaviorally identified by human subjects) in a number of domains. For example, Figure 22 shows three trees over the same object set, but arranged so that human subjects behaviorally identified a different level as basic in each case [HOFF83]. From the top tree down, the basic level is the second, third, and fourth level, respectively (where the root is the first level). Gluck and Corter show that category utility correctly predicts the basic level in each case.

There are several properties of category utility worth mentioning at this point. First, category utility has the desirable property that a partition will score 0 when attribute value distributions are independent of category membership. This indicates that nothing can be inferred from knowing category membership that could not be inferred equally well without such knowledge. Consider the case where an attribute value is independent of category membership:

$$P(A_i = V_{ij} \wedge N_k) = P(A_i = V_{ij})P(N_k).$$

Dividing both sides by $P(N_k)$ gives an alternative definition for independence,

$$P(A_i = V_{ij}|N_k) = P(A_i = V_{ij}).$$

Therefore, if an attribute's value is independent of a category,

$$P(A_i = V_{ij}|N_k)^2 - P(A_i = V_{ij})^2 = 0.$$

If this is the case over all categories and attribute values, it is not difficult to see that category utility equals 0.

While category utility is not a function of attribute-value correlations, it nonetheless tends to favor categories that are formed around such correlations. For example, assume that a category, N_k , is formed around an attribute value,

$A_1 = V_{1j_1}$. That is, $P(A_1 = V_{1j_1}|N_k) > P(A_1 = V_{1j_1})$, which implies $P(A_1 = V_{1j_1}|N_k)^2 - P(A_1 = V_{1j_1})^2 > 0$. Assume that the category utility calculation involves a subexpression of the form

$$P(A_1 = V_{1j_1}|N_k)^2 - P(A_1 = V_{1j_1})^2 + P(A_2 = V_{2j_2}|N_k)^2 - P(A_1 = V_{2j_2})^2.$$

If V_{1j_1} is positively correlated with another value, V_{2j_2} , and N_k captures this correlation, then $P(A_2 = V_{2j_2}|N_k)^2 - P(A_1 = V_{2j_2})^2 > 0$ as well. This increases the category utility value. If V_{1j_1} and V_{2j_2} are independent then we would expect the difference between $P(A_2 = V_{2j_2}|N_k)^2$ and $P(A_1 = V_{2j_2})^2$ to be approximately 0. Thus, categories that capture intercorrelations are rewarded.

Last, while section 3.2 downplayed the importance of nonlinearly separable recognition as applied to solitary concepts, category utility is a quadratic function and does not necessarily favor linearly separable categories. For example, of the two category sets used in experiments by Medin, the nonlinearly separable sets receive a higher category utility score (i.e., $CU = 0.125$) than the linearly separable set (i.e., $CU = 0.1094$).

In conclusion, category utility is a function that rewards intra-category similarity and inter-category dissimilarity. Moreover, category utility can also be viewed as a measure that favors categories that maximize inference abilities. However, rather than maximizing correct prediction of predefined categories – the implied goal of some typicality studies – the measure favors correct prediction across many attributes. Inter-category dissimilarity tends to improve the diagnosticity or predictiveness of attribute values, $P(N_k|A_i = V_{ij})$. Intra-category similarity tends to improve the predictability of attribute values, $P(A_i = V_{ij}|N_k)$, once an object is classified.

Given discussion from chapter 1, these characteristics suggest that category utility may be an ideal measure to guide concept formation. In fact, COBWEB

adopts category utility as a measure of partition quality. Although category utility was developed by assuming extensionally represented categories, it should be clear that category utility can be computed from summary information that includes $P(A_i = V_{ij}|N_k)$ for each value and category. In effect, a summary category representation in the form of a probabilistic concept (along with $P(N_k)$ for each category) is sufficient for computing category utility.

3.3.4 Basic Level and Typicality Effects

Experimental studies illustrate the importance of two dimensions in hierarchical classification schemes [Rosch78]. A *horizontal* dimension is concerned with the way that objects are partitioned into contrasting categories. Typicality studies are fundamentally concerned with this dimension. Classification using competing categories is influenced by the object's typicality with respect to the categories. A *vertical* dimension relates to the arrangement of categories by their generality or inclusiveness. Basic level effects point to a preferred entry point along the vertical dimension. One would expect that these dimensions interact significantly; object set partitions dictate generality relations between categories. Surprisingly, little exploration of the interaction between these dimensions has been carried out, but Jolicour, Gluck, and Kosslyn [JOLI84] are a notable exception.

An important statement of this dissertation is that typicality and basic level effects stem from the same underlying principles of concept structure and organization. For example, while collocation has been proposed as a predictor of basic level concepts, the product of cue and category validity also instantiates Rosch and Mervis' general family resemblance function. Collocation as a measure of typicality is discussed at greater length in chapter 7. An important implication of this analysis is that basic level concepts are the most typical 'images' of superordinate categories.

The weaknesses of independent cue models have motivated many researchers to introduce alternative models of individual concepts. However, section 3.2 showed that another option considered systems (i.e., trees) of concepts. Chapter 7 shows that classification with concept trees accounts for certain typicality and basic level effects. However, it also offers opportunities for studying the interaction between these effects. In particular, the impact of typicality on entry point [JOLI84] is modeled. Second, the idea that typical objects are more quickly recognized in target recognition tasks is hypothesized to be dependent on the level (i.e., superordinate, basic, subordinate) of the target concept.

3.4 Chapter Summary

This chapter developed psychologically motivated constraints on concept induction. The chapter focussed on two lines of experimental study in cognitive psychology, typicality effects and basic level effects. The important conclusions of the chapter include:

- Experimental evidence of typicality in humans suggests that not all concept members are 'created equal'.
- Logical concept representations have difficulty accounting for typicality effects, but probabilistic representations using a summing recognition procedure can account for them.
- Objections to independent cue models stem from naive assumptions about the relation between individual concepts and larger conceptual organizations.
- Probabilistic concept trees overcome the representation problems of individual probabilistic concepts and are informationally equivalent to relational cue and exemplar concept models. In fact, concept trees can be viewed as efficient implementations of these alternative models.
- Experimental evidence of a preferred or basic level in hierarchical classification schemes suggests that recognition cannot occur in a simple top-down fashion. A plausible model of these effects should allow classification to skip levels.

- The identification of category utility as a measure that accurately predicts the basic level makes it a prime candidate for evaluating concept quality during concept induction.

The following chapter describes COBWEB, a system that uses category utility to guide concept formation. Chapters 7 and 8 describe an indexed classification structure that accounts for typicality and basic level effects and suggests a number of further experimental studies into basic level effects, typicality effects, and the interplay between them.

Chapter Acknowledgements

An expansion of a previous draft's discussion of relational cue and exemplar models was motivated by and benefited from discussions with Stephanie Sage, Pat Langley, David Aha, and Dennis Kibler. Comments by Dennis Kibler and David Aha helped the correctness of section 3.3.5. Jeff Schlimmer formatted and provided me with the trees of Figure 22.

CHAPTER 4

COBWEB: An Incremental Conceptual Clustering System

4.1 Chapter Overview

This chapter describes COBWEB, an incremental system for conceptual clustering that brings together aspects of chapters 2 and 3. It builds classification trees where each node is a probabilistic concept. Hill climbing is used in conjunction with category utility to search the space of partitions and classification trees.

Section 2 reviews the environmental and knowledge base assumptions of COBWEB, concentrating on the form of classification trees and the use of probabilistic concepts.

Section 3 gives four operators that are at the core of the COBWEB algorithm. Operator application is controlled so that a hill-climbing search strategy emerges. Moreover, the operator set allows bidirectional movement during search.

Section 4 demonstrates possible interactions between the core operators over multiple observations. This discussion motivates a fifth operator that eases the task of finding good classification trees.

Section 5 presents a way of dealing with missing object information. This simply involves adding an additional weight ('salience') to the category utility evaluation function.

Section 6 gives a way of modifying COBWEB, as developed in earlier sections, to build *superordinate* classes. This discussion is primarily useful for understanding the eventual development of COBWEB/2 which builds superordinate classes in a

	BodyCover	HeartChamber	BodyTemp	Fertilization	Color
mammal-1	hair	four	regulated	internal	brown
reptile-1	corn-skin	imp4	unregulated	internal	green
amphibian-1	moist-skin	three	unregulated	external	red
bird	feathers	four	regulated	internal	red
reptile-2	corn-skin	imp4	unregulated	internal	gray
fish	scales	two	unregulated	external	red
mammal-2	hair	four	regulated	internal	gray
amphibian-2	moist-skin	three	unregulated	external	green

Table 5

Animals described in terms of attribute - value pairs

very natural manner. However, modifying COBWEB to build these classes is somewhat tangential and can be skipped without undue hardship. This modification is not used in future experiments with COBWEB.

Finally, section 7 compares COBWEB's control structure and representations with several other systems.

4.2 Environment and Knowledge Base

COBWEB is an incremental, hierarchical conceptual clustering system. Its function can be stated as:

Given: • a nominal, attribute - value description of an object, o_i , and
 • a classification tree, T , that classifies a set of objects, O .

Find: • a classification tree T' that classifies $\{o_i\} \cup O$.

However, it is often convenient to think of COBWEB's performance only after a significant amount of processing. Beginning with an *empty* classification tree, COBWEB forms a classification tree over a stream of objects. That is,

Given: • a set of object descriptions, $O = \{o_1, \dots, o_n\}$

Find: • a classification tree, T , that classifies all the members of O .

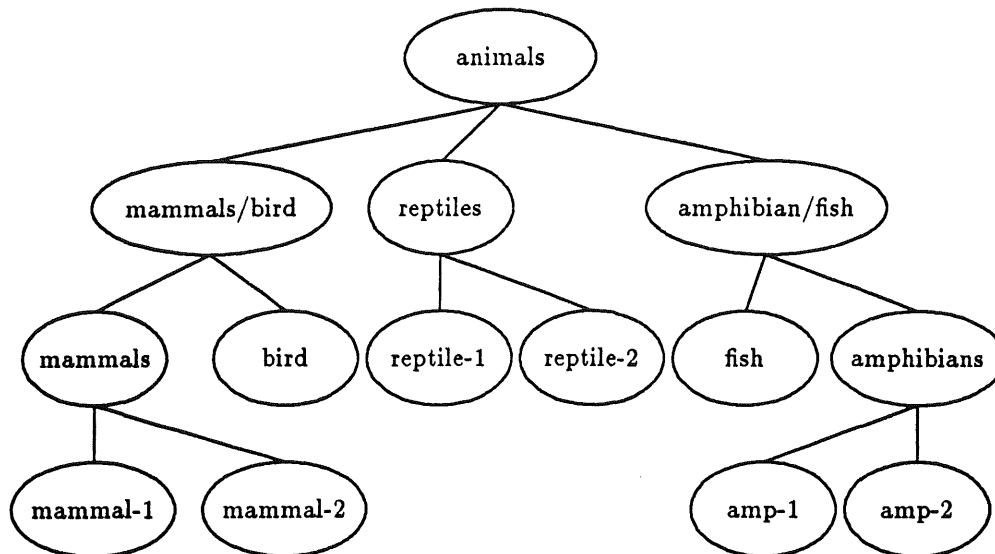


Figure 23

Classification tree over animal descriptions

For example, assume that clustering occurs over the ordering of animals (objects) given in Table 5. Given this data, COBWEB forms the classification tree of Figure 23.²⁸

There are several important aspects of COBWEB classification trees. First, as with other hierarchical systems, COBWEB forms strict trees. There is exactly one path from the root to any leaf. Second, trees have only node-labeling concepts that characterize objects classified under the node. Each node holds a probabilistic concept like those described in chapter 3. However, probabilities are not stored directly. Instead, integer counts are stored and used to compute probabilities as needed. For each node there is a count of the number of objects classified under the node, as well as a count of the objects stored under the node with each attribute value. From these counts it is possible to compute the probability of an object having a particular value. For example, consider the expansion of the 'animals'

²⁸ In general, different orderings will result in different classifications. However, COBWEB will tend to converge on the same tree regardless of input order. Convergence is explored in Chapter 6.

$$P(\text{animal}) = 1.0$$

$$P(\text{BodyCover} = \text{hair} \mid \text{animal}) = 0.25$$

$$P(\text{BodyCover} = \text{feathers} \mid \text{animal}) = 0.125$$

$$P(\text{BodyCover} = \text{cornified-skin} \mid \text{animal}) = 0.25$$

$$P(\text{BodyCover} = \text{moist-skin} \mid \text{animal}) = 0.25$$

$$P(\text{BodyCover} = \text{scales} \mid \text{animal}) = 0.125$$

$$P(\text{HeartChamber} = \text{four} \mid \text{animal}) = 0.375$$

$$P(\text{HeartChamber} = \text{imperfect-4} \mid \text{animal}) = 0.25$$

$$P(\text{HeartChamber} = \text{three} \mid \text{animal}) = 0.25$$

$$P(\text{HeartChamber} = \text{two} \mid \text{animal}) = 0.125$$

$$P(\text{BodyTemp} = \text{regulated} \mid \text{animal}) = 0.37$$

$$P(\text{BodyTemp} = \text{unregulated} \mid \text{animal}) = 0.63$$

$$P(\text{Fertilization} = \text{internal} \mid \text{animal}) = 0.63$$

$$P(\text{Fertilization} = \text{external} \mid \text{animal}) = 0.37$$

$$P(\text{Color} = \text{brown} \mid \text{animal}) = 0.125$$

$$P(\text{Color} = \text{gray} \mid \text{animal}) = 0.25$$

$$P(\text{Color} = \text{red} \mid \text{animal}) = 0.375$$

$$P(\text{Color} = \text{green} \mid \text{animal}) = 0.25$$

Table 6

Expansion of the "animals" node

node in Table 6. The root node shows that 25% of all observed objects have hair. The quotient of the number of objects classified at the root that have hair (2) and the number of all objects classified at the root (8) equals 0.25. That is, $P(\text{Body-cover} = \text{hair} \mid \text{animals}) = 0.25$.

A second measure associated with each node is the probability of the node itself. This is the proportion of objects classified under the node that are also classified under the node's parent. For example, since all observed objects are classified under the root (the root has no parent), the probability of an arbitrary observed object being classified under it is 1.0. As a second example, consider the expanded "mammal" node of Figure 23 in Table 7. Associated with this node is

$$P(\text{mammal}|\text{mammal/bird}) = 0.67$$

$$P(\text{BodyCover} = \text{hair} \mid \text{mammal}) = 1.0$$

$$P(\text{HeartChamber} = \text{four} \mid \text{mammal}) = 1.0$$

$$P(\text{BodyTemp} = \text{regulated} \mid \text{mammal}) = 1.0$$

$$P(\text{Fertilization} = \text{internal} \mid \text{mammal}) = 1.0$$

$$P(\text{Color} = \text{brown} \mid \text{mammal}) = 0.5$$

$$P(\text{Color} = \text{brown} \mid \text{mammal}) = 0.5$$

Table 7

An expansion of the "mammal" node

the probability of the node *in the context of* the "mammal/bird" node. That is, 67% of the objects classified under "mammal/bird" are classified as mammals.

4.3 COBWEB's Basic Control Structure

COBWEB is an incremental system for hierarchical conceptual clustering. In accordance with chapter 2, COBWEB can be viewed in terms of search. Heuristic search requires attention to several issues:

- a *state representation, initial and goal states,*
- a *heuristic evaluation function* used to guide search,
- *operators* used to explore the state space, and
- a *control strategy* that coordinates operator application.

To review discussion from chapter 2, conceptual clustering proceeds at three levels of search: through concepts, partitions, and hierarchies. Each of these spaces has its own state representations, heuristics, operators, and control strategies. The bottom-most search for descriptive concepts is the task of learning from examples. The second level of search is for partitions of the observed objects. In principle, any partition of the total set can be viewed as the goal state, but in practice a heuristic measure directs search to preferred or optimal partitions. Last, a search through hierarchies uses the subordinate processes of finding concepts and partitions to

build classification hierarchies. Like the search for partitions, any classification tree can be viewed as a goal, but conceptual clustering systems typically use a measure of tree quality to explicitly guide search [LANG84] or rely on heuristics of subordinate searches to implicitly define tree optimality [FIS85A, MIC83A].

As a hierarchical conceptual clustering system, COBWEB can be viewed as conducting a search at three levels. COBWEB classification trees and probabilistic concepts have just been described. In addition, the first level of a classification tree gives the partition of the entire set of observed objects that is found by COBWEB. Before describing the COBWEB algorithm in detail, it is useful to briefly consider some high-level characteristics of the searches through these three spaces.

4.3.1 Overview of COBWEB's Search Processes

The bottom most level of search is the one for concept descriptions. The probabilistic concepts of COBWEB list all and only attribute values that are present in one or more observed instances. Additionally, counts of the number of class members with each attribute are listed, as well as the total number of class members. The sole operator used by COBWEB to update a concept description based on a new object is to union in the attribute values of the object and to update corresponding counts as well. Class and attribute value counts (and probabilities computed from these counts) are completely determined by the members of the class. There is exactly one probabilistic concept for any given class. As with the maximally-specific attribute value concepts of chapter 2, COBWEB's 'search' for probabilistic concepts is deterministic.

The second level of search is the one through a space of object set partitions. Chapter 2 indicated that there were two points of possible nondeterminism in this search: 1) which class of several possibilities would an object be placed and 2) which concept description would be selected to describe the updated class. However, discussion above points out that selecting a concept description is not

a point of nondeterminism. In COBWEB the only decision that need be made in searching partitions is which class of the a partition should incorporate a new object. Category utility [GLUC85], the measure of partition quality developed in chapter 3, is used to decide upon *one* class to add the object. This results in a single new partition. Thus, COBWEB hill climbs through the space of partitions.

The final level of search is through a space of classification trees. The nondeterminism of this search is restricted by the search for partitions. Compositions of the partition-update operators, one per tree level, as one descends a classification tree can be viewed as transforming the tree in one way. The emergent search strategy is one of hill climbing through a space of classification trees. COBWEB does not use an explicit measure of classification tree quality to guide this search. Rather, tree quality may be defined lexicographically, in terms of the quality of its constituent partitions; an optimal tree is one in which the first level optimally partitions the observed objects (according to category utility) and each subtree is optimal in the same manner with respect to the subset of objects that it classifies.

In summary, the search for concepts is deterministic in COBWEB and the search through the space of hierarchies is completely dictated by the search for partitions. This search for partitions is heuristically guided by category utility; the objective of this search is the partition that best divides the observed objects according to category utility.

The remainder of this section explains the operators used to search the space of partitions. Object incorporation is basically a process of classifying the object by descending the tree along an appropriate path. At each tree level one of several possible operators is applied. These operators are

- classifying the object with respect to an existing class,
- creating a new class,
- combining classes into a single class, and
- dividing a class into several classes.

These operators are applied to a single object set partition (set of siblings in the tree).

4.3.2 Operator 1: Placing an Object in an Existing Class

Perhaps the most natural way of updating a partition of objects when a new object is observed is to place the object in an existing class. That is, after updating the counts of attribute values at the root of a hierarchy, the object is incorporated into one of the root's children. To determine which child 'best' hosts a new object, the object is tentatively placed in each child and appropriate counts of the node are temporarily updated. The partition that results from adding the object to a given node is evaluated using category utility (function 3-4). The node that results in the best partition is chosen to assimilate the new object and corresponding counts are permanently updated. For example, if a partition consists of nodes corresponding to 'mammals', 'birds', and 'reptiles', a new animal description, 'dog-1', would be added to the 'mammals' class iff $CU(\text{'mammals'+ 'dog-1', 'birds', 'reptiles'}) > CU(\text{'mammals', 'birds'+ 'dog-1', 'reptiles'})$ and $CU(\text{'mammals'+ 'dog-1', 'birds', 'reptiles'}) > CU(\text{'mammals', 'birds', 'reptiles'+ 'dog-1'})$.

Recursively applying this operator to best host nodes causes a descent of the classification tree that eventually bottoms out at a leaf. This descent can be viewed as a recognition process for the newly observed object.

4.3.3 Operator 2: Creating a New Class

In addition to placing objects in existing classes, there is a way of creating new classes. After the best host among the existing classes has been determined, the quality of the *partition* resulting from placing the object in the best existing host is compared to the partition that results from creating a new singleton class for the object. Depending on which partition is best (with respect to category utility), the object is either placed in the best existing class or a new class

is created. For example, if an object, 'fish-1', is encountered, and 'reptiles' is the best existing host among 'mammals', 'birds', and 'reptiles', then a singleton class, 'fishes' \equiv {fish-1} is created iff $CU(\text{'mammals', 'birds', 'reptiles', 'fishes'}) > CU(\text{'mammals', 'birds', 'reptiles'} + \text{'fish-1'})$.

This operator allows COBWEB to adjust the number of classes at a partition to fit the regularities of the environment. This ability distinguishes COBWEB from methods that depend on a system parameter to decide how many classes are to be formed (e.g. CLUSTER/2).

4.3.4 A Simple Example

Figure 24 demonstrates the effect of operators 1 and 2. Snapshot (a) shows a classification tree that has been previously built over the 'fish' and 'amphibian-2' objects of Table 5. Listed with each node (class) are the probability of the class and the probabilities of attribute values conditioned on class membership. For example, the probability of having scales is 0.5 for objects classified at the root of snapshot (a), while scales are assured with probability 1.0 for objects classified at N_1 (a singleton class containing only 'fish'). Space has allowed showing only one attribute value for each node, but *all* values exhibited over objects of a node are stored with their respective conditional probabilities (e.g., like the expanded nodes of Tables 6 and 7). Last, probabilities reflect attribute-value distributions over observed objects. As with any inductive program there is an implicit assumption that the observations collectively approximate the environment as a whole. However, distributions are not permanent, but change in response to further observation [CHEE85].

Snapshot (b) shows a new class being created. The transition from (a) to (b) is caused by incorporating the 'mammal-1' object of Table 5. The probability, $P(\text{scales} | N_0)$, reflects this addition at the root. Creating a new singleton class

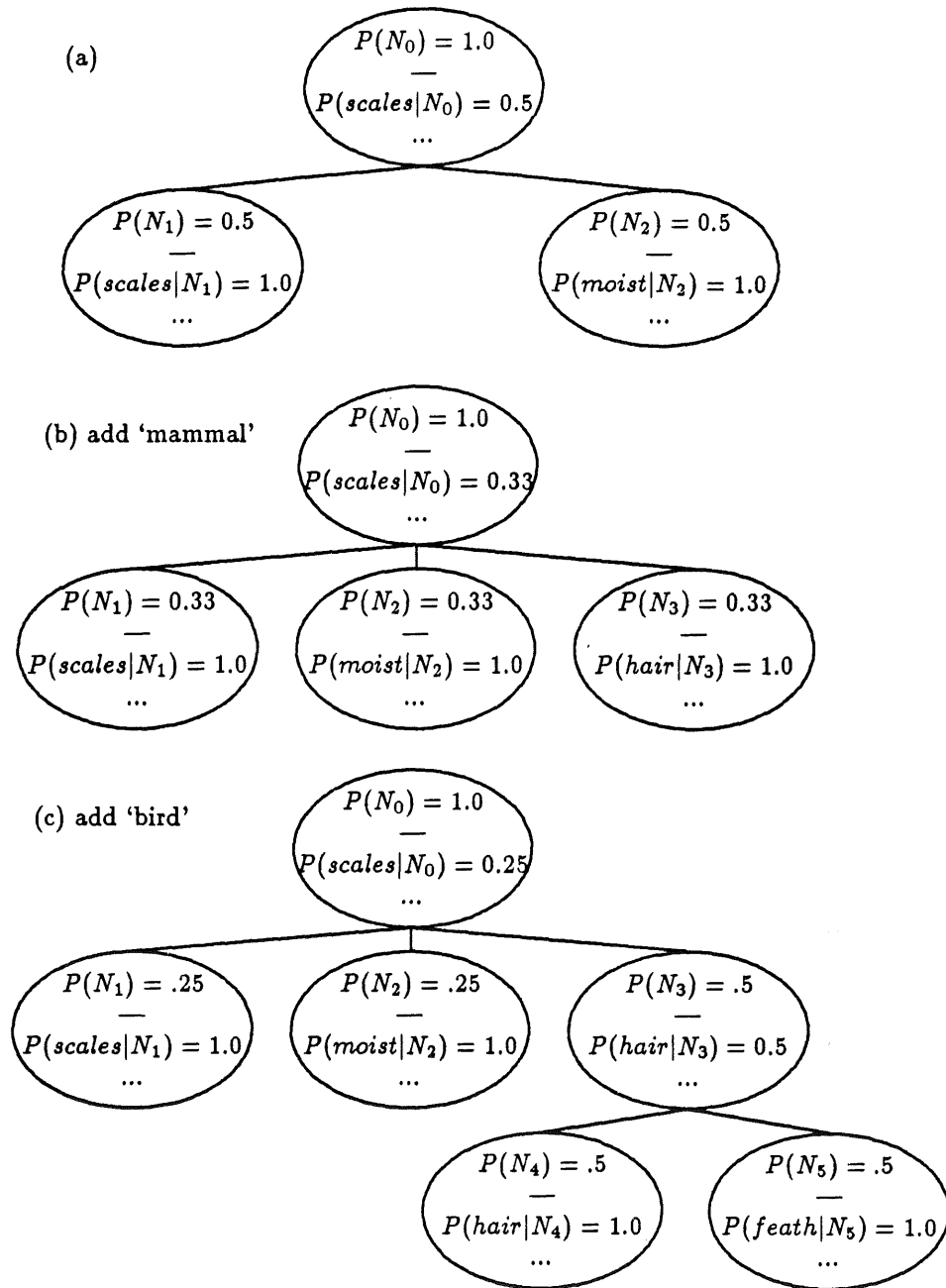


Figure 24

Adding 'mammal' and 'bird' to memory

(N_3) corresponding to 'mammal' yields a better partition than adding the object to either of the existing classes.

Snapshot (c) demonstrates an object being added to an existing class. Adding 'bird' to the tree of snapshot (b) causes appropriate alterations at the root, e.g., scales now occur in only one quarter of the observed animals. Adding 'bird' to the existing class corresponding to 'mammal' yields the best possible partition. Since this node is a leaf in snapshot (b), incorporation of 'bird' involves expanding the leaf to accommodate the new object, as well as the previously classified one. Leaf expansion occurs whenever an incoming object differs from the object stored at the leaf.²⁹

Figure 24 demonstrates how a classification is constructed over sequential observations and how distributions change to reflect increasing information. While this figure shows probabilities at each node, recall that they are actually computed from integer counts. Stored at each node is a count of the number of objects classified under the node and each attribute-value entry includes an integer count of the number of objects classified under the node possessing that value. From these counts the probability of a value conditioned on class membership can be computed. Probabilities are computed on demand for evaluation purposes, but it is the underlying counts that are updated.

4.3.5 Operators 3 and 4: Merging and Splitting

While operators 1 and 2 are effective in many cases, by themselves they are very sensitive to initial input ordering. Although a classification tree should reflect structure inherent in the environment, skewed data may present a different image of that structure than representative data. So far there are no mechanisms for adjusting a classification tree when initial observations prove unrepresentative.

In order to mitigate the effects of skewed data, COBWEB includes operators for node *merging* and *splitting*. Merging combines two nodes of a level (of n nodes)

²⁹ COBWEB remembers every observed object as a leaf in the classification tree. However, variations are possible in which instances are 'forgotten' or dropped from the tree.

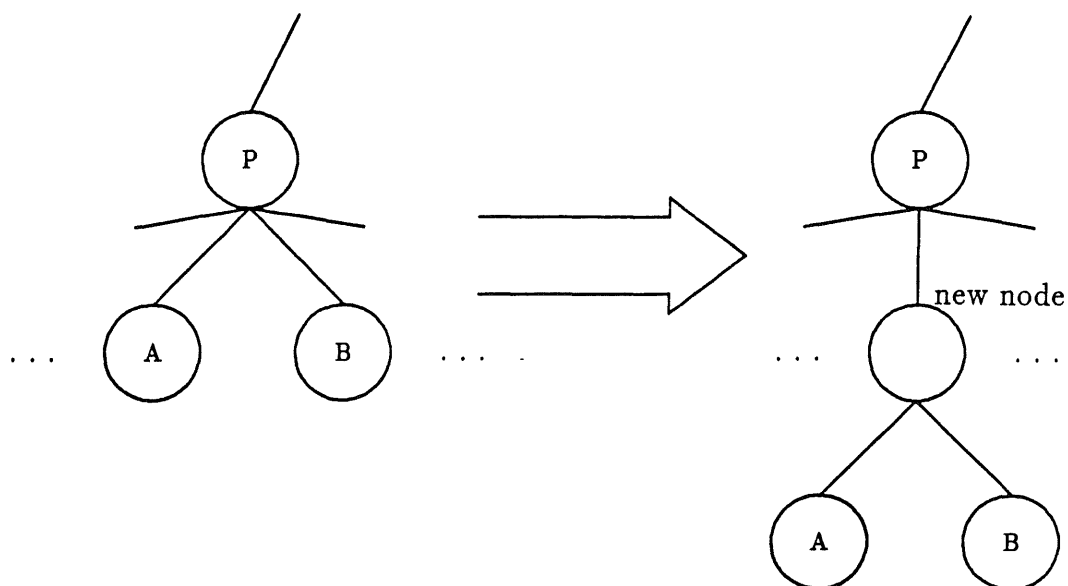


Figure 25

The effect of node merging

in hopes that the resultant partition (of $n - 1$ nodes) is of better quality. Merging two nodes involves creating a new node and summing the attribute value counts of the nodes being merged. The original nodes are made children of the newly created node.

Although merging could be attempted on all possible node pairs every time an object is observed, this strategy is unnecessarily redundant and costly. Instead, when an object is incorporated, only merging the *two* best hosts (as indicated by category utility) is evaluated. Figure 25 illustrates the general form of merging; if classes 'A' and 'B' are the first and second best hosts for a new object, 'O', then merging 'A' and 'B' occurs iff $CU('A'+'B'+'O', \dots) > CU('A'+'O', 'B', \dots)$.

A more specific example of the effect of merging is shown by the tree of Figure 26. This tree results if 'amphibian-1' is added to the tree of Figure 24c. This addition causes N_1 and N_2 to be merged into a single node N_6 . This new node, once updated to reflect the addition of 'amphibian-1' classifies 60% of the

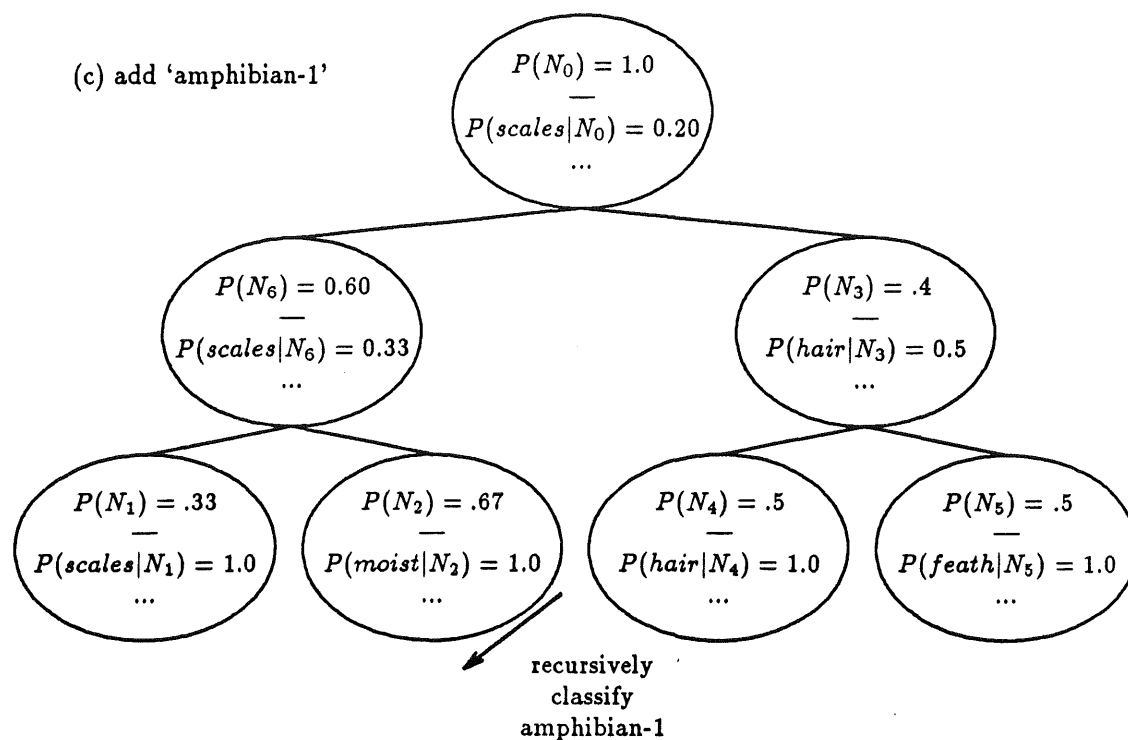


Figure 26

Merging caused by adding 'amphibian-1'

total objects. Furthermore, 33% of the objects classified under N_6 have scales, while 67% have moist skin.

The effect of merging the two best hosts is that changes in the tree are restricted to areas activated during object recognition. This locality constraint is enforced, albeit in different forms, by systems such as UNIMEM and CYRUS, and is behind Schank's ideas of *reminding* and dynamic memory [SCHA82]. Selecting only two nodes to merge (rather than three, four, or more) is the cheapest way of introducing more general classes. Interactions with other operators insures the generality of the approach. This is discussed in section 4.

In addition to merging, node splitting may serve to increase partition quality. If a node carries little information, it may be desirable to split it up so that subclasses can be reorganized into a more cohesive unit. In general, there are many

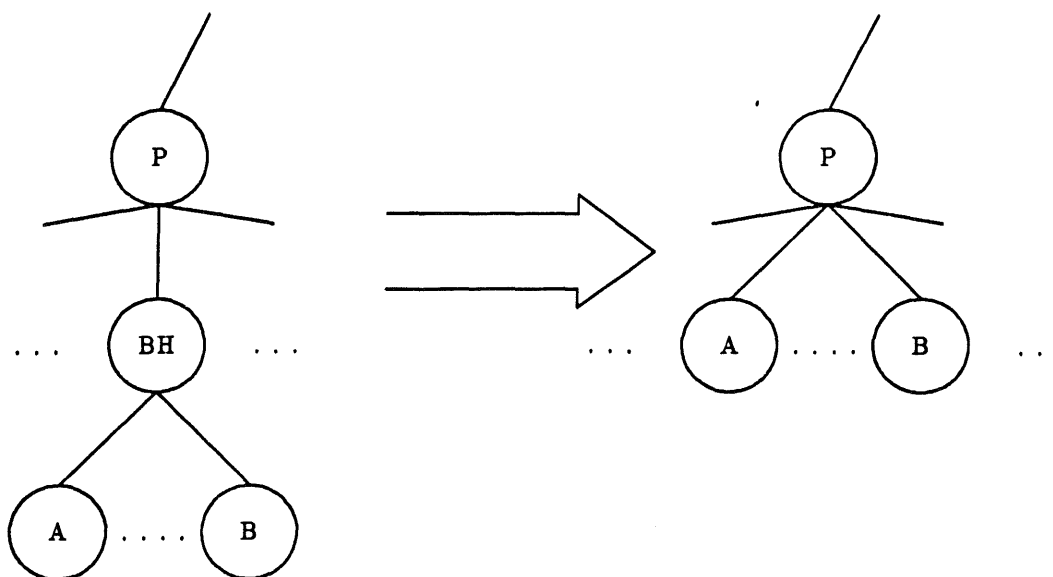


Figure 27

The effect of node splitting

ways to split a class up, but only one is tried. A node is split up into subclasses corresponding to the node's children, since these children tend to represent the 'best' partition of objects classified under the node. Splitting a node of a partition (of n nodes) is performed by deleting the node and promoting its children. This results in a partition of $n + m - 1$ nodes, where the deleted node had m children. Thus, this operator involves *deleting* a node of the tree, but looking at this as an operator on partitions, it has the appearance of *splitting* a node into subclasses.

Besides restricting the ways in which a node can be split, there is also a restriction on what nodes to consider splitting. Splitting is considered only for the best host among the existing categories. For example, assume 'BH' is the best host for an object, 'O'. Additionally, 'BH' has a number of children, including 'A' and 'B'. Of these, 'A' is the best host of 'O' among 'BH's children. 'BH' is split and replaced by its children iff $CU(\{'A' + 'O', 'B', \dots\}) > CU(\{'BH' + 'O', \dots\})$. Thus, splitting requires looking ahead one level. The general form of splitting is illustrated in Figure 27.

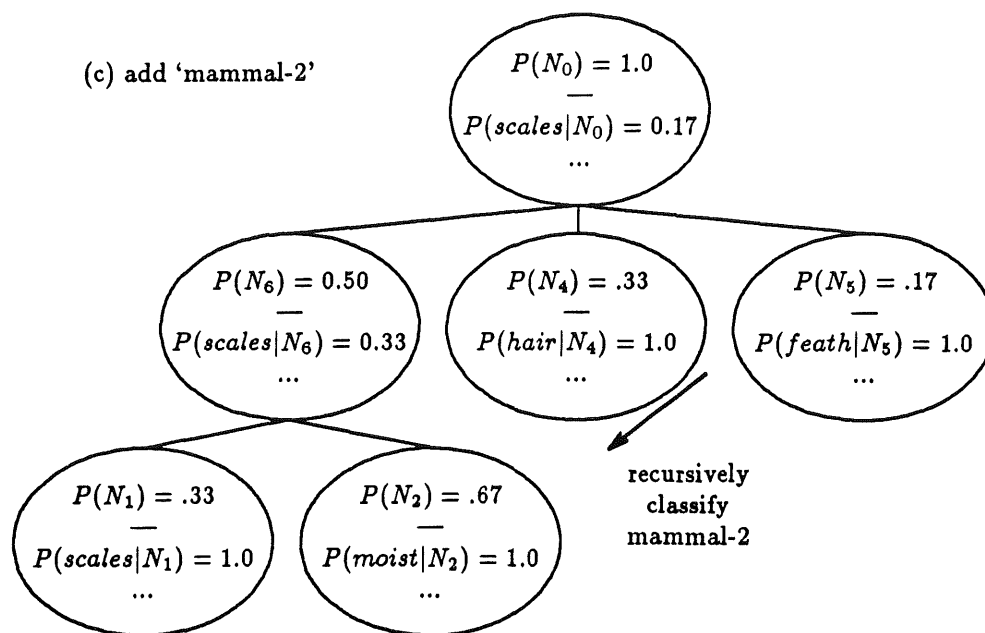


Figure 28

Splitting caused by adding 'mammal-2'

An example of node splitting is shown in Figure 28. The tree of this figure assumes that 'mammal-2' from Table 5 has been added to the tree of Figure 26. Node N_3 has been split and 'mammal-2' has been recursively classified with respect to N_4 . Splitting N_3 assumes that to do so increases the quality of the top-most partition. However, this is not true in the case of this tree. In reality, N_3 would not have been split when adding 'mammal-2'. The purpose of Figure 28 is to show the effect of splitting, *if* it were called for on the previously developed example.

Merging and splitting are roughly inverse operators, but not perfectly so, since merging combines exactly two nodes, while splitting promotes two or more nodes, depending on how many children a split node has. However, they do allow COBWEB to move bidirectionally through a space of possible hierarchies. Splitting can undo the effects of a prior merge and vice versa. In general, node merging is invoked when initial observations suggest that the environment is a space of

FUNCTION COBWEB (Object, Root (of a classification tree))

- 1) Update counts of the Root
- 2) IF Root is a leaf THEN Return expanded leaf to accommodate object
 ELSE Find that child of Root which best hosts Object and perform
one of the following
 - 2a) Consider creating a new class and do so if appropriate
 - 2b) Consider node merging and do so if appropriate and call
 COBWEB (Object, Merged node)
 - 2c) Consider node splitting and do so if appropriate and call
 COBWEB (Object, Root)
 - 2d) IF none of the above (2a,b, or c) were performed THEN call
 COBWEB (Object, Best child of Root)

Table 8

The control structure of COBWEB

highly similar objects, relative to the actual structure of the environment suggested by subsequent observations. Splitting is invoked when the environment is more 'compressed' than suggested by initial input.

4.3.6 Control of COBWEB's Four Operators

Table 8 shows COBWEB's control of the four operators described so far. These operators share the assumption that change in a classification tree is localized around areas 'activated' during recognition. Additionally, for each object incorporated, only one operator can be applied at a given level. Thus, each object serves to transform a single partition into a single new partition. Taken collectively, these changes transform a single tree into a single new tree. Therefore, COBWEB is hill climbing through the space of classification trees. The objective of this search is to obtain a tree that optimally partitions the observations at each tree level according to category utility. An alternative interpretation is that COBWEB's objective is to search for a single optimal partition. The top-most level is the best hypothesis. Subtrees are heuristically ordered sub-partitions that can be weaved into the top partition (via splitting) as it is deemed necessary.

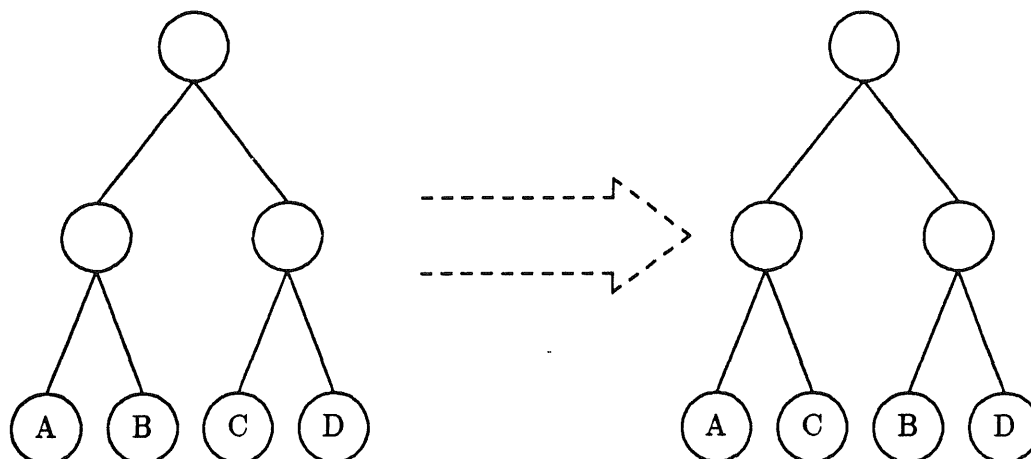


Figure 29

Tree transformation by applying *split*, *split*, *merge*, *merge*

Regardless of the interpretation, COBWEB can be viewed as hill climbing with bidirectional operators. Chapter 6 experimentally demonstrates some of the limitations inherent in this strategy. The following section looks at some example operator interactions over multiple object incorporations. This discussion gives an analytic, but intuitive, picture of hill-climbing weaknesses and strengths, as well as motivating a fifth operator called *promotion*.

4.4 Operator Interactions

In principle, the four operators of (1) classification, (2) creation, (3) merging, and (4) splitting are sufficient for constructing a classification tree of any form. Given sufficient observations, these operators can completely traverse the space of classification trees. For example, consider how merging and splitting could be combined to bring about the transformation of Figure 29. Ideally, these operators should interact so as to guarantee that given sufficient observations, a classification tree is found that optimally partitions observations at each level. However, preconditions for applying these operators are restrictive. They may be applied only if to do so immediately improves the quality of a partition. The limited

foresight of operator preconditions can hinder transformations that are necessary for optimality.

One way to guarantee optimal partitions is to check whether operator combinations (e.g., split-merge, classify-split-merge) improve partition quality. Applying 'macro-operators' would guarantee optimal object partitions, but would require a search that would greatly cripple any claim that COBWEB was a useful incremental system. This search could be reduced by bounding the size of macro-operators. For instance, COBWEB might only examine macro-operators of size two (e.g., split-split), as well as examining the effects of the solitary operators. However, even this scheme involves considerable search, and it would not eliminate the problem it had been designed to solve.

Rather than guaranteeing optimal behavior, COBWEB's integrity as a hill climber is maintained. Finding good classifications is dependent on indirect interaction between operators that are independently applied over multiple observations. The ramifications of this strategy are experimentally demonstrated in chapter 6. However, it is helpful to describe some cases of operator interaction and show how they impact tree organization.

4.4.1 Creating New Classes

Medin [MEDI83] points out that many similarity measures used in concept formation do not adequately distinguish when a new class should be created. The problem arises because these measures favor classifying an object with respect to an existing category as existing categories increase in size. Consider the simple case where two categories exist, N_1 and N_2 . Suppose they contain m_1 and m_2 objects, respectively, where $m_1 + m_2 = m - 1$. If category utility is used to evaluate partition quality then $P(N_k) = \frac{m_k}{m-1}$ and $CU(\{N_1, N_2\})$ equals

$$\frac{\frac{m_1}{m-1} \sum_i \sum_j P(A_i = V_{ij} | N_1)^2}{2} + \frac{\frac{m_2}{m-1} \sum_i \sum_j P(A_i = V_{ij} | N_2)^2}{2} - \frac{\sum_i \sum_j P(A_i = V_{ij})^2}{2}$$

A new object will be evaluated with respect to N_1 and N_2 , as well as checking whether a new category should be created. A partition with a newly created category, N_3 will have a $CU(\{N_1, N_2, N_3\})$ equal to

$$\frac{\frac{m_1}{m} \sum_i \sum_j P(A_i = V_{ij} | N_1)^2}{3} + \frac{\frac{m_2}{m} \sum_i \sum_j P(A_i = V_{ij} | N_2)^2}{3} + \frac{\frac{1}{m} \sum_i \sum_j P(A_i = V_{ij} | N_3)^2}{3} - \frac{\sum_i \sum_j P(A_i = V_{ij})^2}{3}$$

As m increases, the probability of a new singleton category, $\frac{1}{m}$, approaches 0.0. For large enough m , the component score for the third category will effectively be 0.0, thus reducing the category utility score (since it is an average over categories). There is a point at which, regardless of the extent to which an object differs from existing categories, it will be forced into one of them.

Despite the apparent limitation on new class creation, the problem is significantly mitigated by considering the interaction of the classification and splitting operators. If an exceptional object is forced into an existing category, similar forthcoming objects will tend to be placed in that category as well. After a sufficient number of these similar objects have been processed, a split will break the more recently formed subclass from the original existing class.

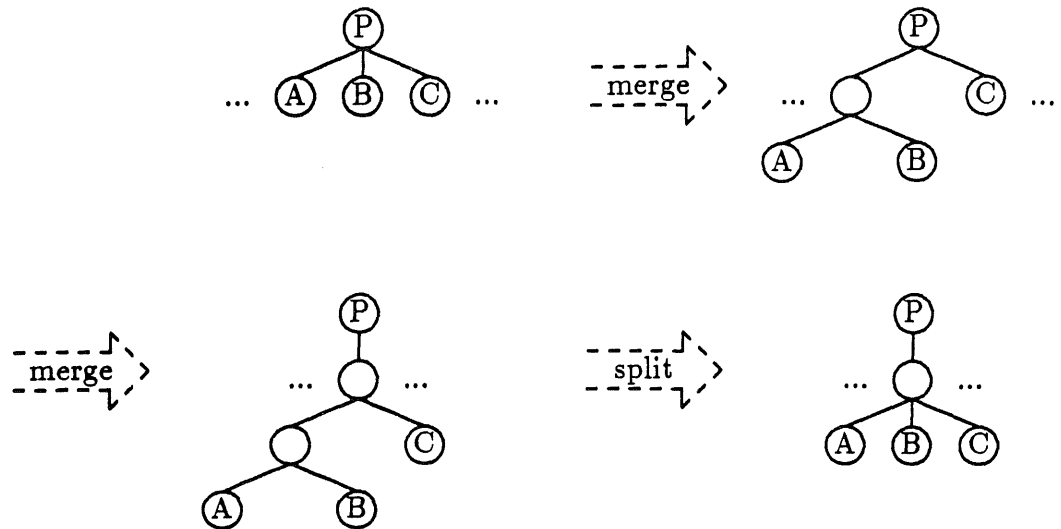


Figure 30

A demonstration of three nodes being 'merged' over multiple trials.

4.4.2 'Merging' More Than Two Nodes

COBWEB allows only two nodes to be merged at a time. However, more than two siblings can be combined through an indirect interaction of merging and splitting over several incorporation trials. Figure 30 demonstrates how three siblings can be 'merged' after several trials. This demonstrates that allowing only two nodes to be merged at a time does not restrict the form of classification trees that can be formed, given sufficient observations.

4.4.3 Useless Nodes and Node Promotion

Situations occur when merging two nodes would result in an optimal partition. However, they cannot be merged because they are not siblings, e.g., suppose one is the uncle of the other. This problem can be generalized and is pictured in Figure 31. Merging A with B (and removing B from its current ancestral line) would improve quality, but such a move requires foresight beyond the effect of single merge or split operators. No amount of (current) operator interaction will alleviate this problem. Rather than implementing the macro facility discussed earlier, COBWEB tries to

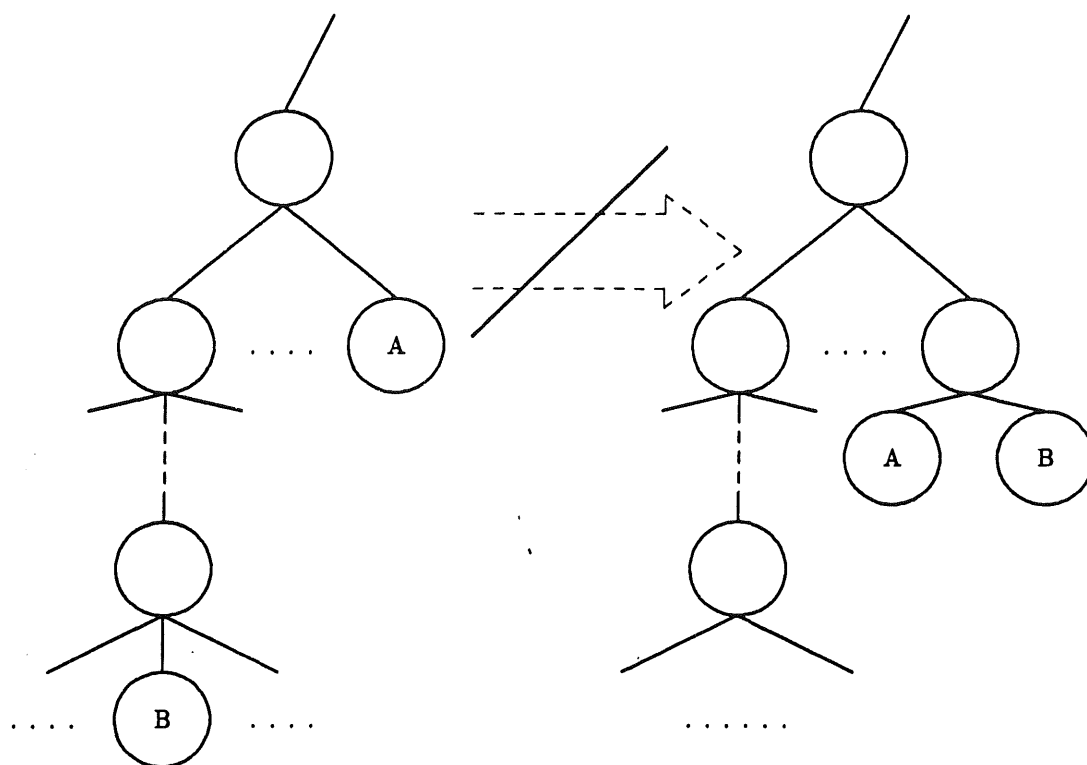


Figure 31

An example of a lack of heuristic 'foresight'

recognize when a node is misplaced and selectively *promotes* it to a level where it can be merged into the 'correct' subtree.

A simple way of determining if a node is misplaced is to climb the classification hierarchy, at each level seeing if the node in question belongs with its current ancestor or with a sibling of its current ancestor. This involves extracting the node from its current ancestor (by decrementing appropriate counts) and alternatively incorporating the node in competing nodes (by incrementing appropriate counts). Evaluating the resultant partitions (by category utility) indicates the best placement of the node. In fact, the gist of this procedure is followed in COBWEB. Notice that this strategy requires parent or IS-A links be added to classification trees.

Probabilities of concept N_k	Prototype of N_k
$P(\text{Color} = \text{red} N_k) = 0.67$	Color = red
$P(\text{Color} = \text{blue} N_k) = 0.33$	
$P(\text{Shape} = \text{square} N_k) = 1.0$	Shape = square
$P(\text{Size} = \text{large} N_k) = 0.5$	
$P(\text{Size} = \text{small} N_k) = 0.5$	Size = small (choice arbitrary)

Figure 32

A node and its prototype used to test for node misplacement

Rather than comparing a node with its ancestors, a *prototype* of the node is created and used for comparative purposes. The prototype of a node is an object description that represents the 'central tendency' of the node. Specifically, a node's prototype has the most frequently occurring attribute values of the node. Figure 32 shows the attribute-value probabilities of a node, and the resultant prototype. A node's prototype is compared to the node as well as the node's siblings. The comparison process is like that described earlier; the object is added successively to each node of a level and the resultant partition is evaluated with respect to category utility. This process may be recursively applied for each of the node's ancestors (and their siblings) until the root of the classification tree is reached.

If a node's prototype matches a nonancestral node better than the node's applicable ancestor, this indicates that the node under scrutiny is probably (but not assuredly) *useless*. That is, if its prototype cannot be classified with respect to its ancestor (and thus the node itself), probably no objects will be. Uselessness is the criterion for determining whether a node should be promoted. Pseudo-code for the uselessness test is given as function the 'USELESS' in Table 9.

To test each node for uselessness would be costly. Therefore, COBWEB employs a cheap method of filtering nodes. Associated with each node is a count of

FUNCTION USELESS (Node-A, Prototype)

IF Node-A is the Root (i.e., no parent) THEN Return false (useful)
 ELSE 1) Classify Prototype with respect to the Parent of Node-A and
 determine the Best-host for Prototype.
 2) IF Best-host \neq Node-A THEN Return true (useless)
 ELSE Return USELESS (Parent of Node-A, Prototype).

Note that Prototype is the prototype of the original node being tested for useless, and does not change during recursive calls.

Table 9

 Pseudo-code for uselessness

the number of trials since the node last classified an object. Every time an object is classified with respect to a node's parent, but is then classified with respect to one of the node's siblings, the 'last-accessed' count of the node is incremented by one.

If a node, N_k , has probability, $P(N_k)$ (with respect to its siblings), then $(1 - P(N_k))^m$ gives the probability (given a random sampling) that a node will *not* classify a single object after m trials. For instance, if a node has a probability of 0.4, the probability of it not classifying an object after five trials is given as $(1 - 0.4)^5 = 0.6^5 \approx 0.078$. This approaches 0.0 as the number of trials (m) since the node's last access grows. In COBWEB, if this probability becomes sufficiently small (0.05 or less), it is an initial indication that the node is useless. Function USELESS is applied to make a final determination. If the node is in fact reachable, the 'last-accessed' count, m , is reinitialized to zero. Reinitialization also occurs when an object is classified with respect to a node.

Once COBWEB has been determined that a node is useless, it may promote the node. Rather than promoting a node to the level that its prototype indicated uselessness, a more conservative approach is taken. A node that is to be promoted may have children. Just as the node may be misplaced with respect to its siblings,

```

PROCEDURE PROMOTE (Node)
IF Node is unreachable with respect to immediate siblings
  THEN Split Node
  ELSE Remove Node as a child of its current parent and
        make it a child of its old grandparent
        Split Node in its current position

```

Table 10

Pseudo-code for promotion

a child of the node may also be misplaced with respect to other children of the node. The approach that empirically seems to yield the best results promotes the useless node a *single* level and then splits it. This allows the node's children to be processed separately when future objects are assimilated.

As new objects are assimilated, useless nodes are successively promoted and split. This promoting and splitting may open the way for the standard merge and split operators. However, a node may be repeatedly (promoted and) split, eventually causing leaves (singleton classes) to surface that represent individual objects. An useless leaf cannot be further split. If it has been promoted as far as possible, it is removed from the tree and reclassified. Reclassification is performed in exactly the same way that any object is classified with respect to a classification tree.

Table 11 gives the pseudo-code for the COBWEB system, including tests for uselessness (and associated test of the last-accessed counter) and promotion. These changes are made after the object has been classified with respect to lower levels. Importantly, node promotion does not assure a globally optimal partitioning of objects. Suboptimal partitions do not necessitate that certain subnodes will become useless. However, in cases where nodes do become useless, this extension insures they will be spotted. Node promotion guarantees locally optimal partitions are obtained over sufficient observations.

FUNCTION COBWEB (Object, Root { of a classification tree })

- 1) Update counts of the Root
- 2) IF Root is a leaf THEN Return expanded leaf to accommodate object
 ELSE Find that child of Root which best hosts Object and perform
one of the following
 - 2a) Consider creating a new class and do so if appropriate
 - 2b) Consider node merging and do so if appropriate and call
 COBWEB (Object, Merged node)
 - 2c) Consider node splitting and do so if appropriate and call
 COBWEB (Object, Root)
 - 2d) IF none of the above (2a,b, or c) were performed then call
 COBWEB (Object, Best child of Root) and
 reinitialize last-accessed counter of Best child.
- 3) FOR each child, N, of Root DO
 IF N has not been accessed in a long while
 THEN IF USELESS (N, Prototype of N)
 THEN PROMOTE(N) { via side effects }
 ELSE reinitialize last-accessed counter of N.

Table 11

Pseudo-code for COBWEB with the promotion operator

4.5 Dealing with Missing Information

COBWEB, as described so far, does not deal with partial object information. So that objects with missing attribute values can be assimilated during tree construction, the category utility measure is extended. This extension adds a *salience* weight to category utility.

Salience is generally regarded as a measure of the 'perceivability' of an object property [SMIT81]. For instance, the color of an object may be considered more salient than the smell. However, COBWEB is not directly concerned with the perception of objects, but only assumes post-perception object descriptions in the guise of attribute-value pairs. Under this assumption, the salience of an attribute will be the probability that the attribute's value is observed. For example, in a sample of 10 objects, if a value for the attribute Color is present in 7 object descriptions, then

$$\text{Saliency}(\text{Color}) \equiv P(\text{Color observed}) = \frac{7}{10}.$$

Saliency can be computed by adding additional integer counters to each node of a hierarchy, one counter for each attribute. These reflect the number of times a value for a given attribute, A_i , was observed. The probability of observing a value of an attribute, A_i , in an object classified under a node, N_k , is

$$P(A_i \text{ observed} | N_k) = \frac{\# \text{ of objects under } N_k \text{ with observed value of } A_i}{\# \text{ of objects under } N_k}.$$

Category utility is extended so that $CU(N_1, \dots, N_n)$ equals

$$\frac{[\sum_{k=1}^n P(N_k) \sum_i P(A_i \text{ observed} | N_k) \sum_j P(A_i = V_{ij} | N_k)^2]}{n} - \frac{\sum_i P(A_i \text{ observed}) \sum_j P(A_i = V_{ij})^2}{n} \quad (4-1)$$

The computation of $P(A_i = V_{ij})$ and $P(A_i = V_{ij} | N_k)$ are also altered to take into account the possibility of incomplete object descriptions. Specifically,

$$P(A_i = V_{ij} | N_k) = \frac{\# \text{ of objects under } N_k \text{ with value } V_{ij} \text{ of } A_i}{\# \text{ of objects under } N_k \text{ with observed value of } A_i}$$

$P(A_i = V_{ij})$ is similarly computed from a count of the number of total objects that have an observed value for attribute A_i . These computations assume that the distribution of observed values reflects the distribution over all (observed and unobserved) values.

Function 4-1 is equivalent to category utility (3-7) if there is no missing information. This follows because $P(A_i \text{ observed})$ and $P(A_i \text{ observed} | N_k)$ will

equal 1.0 for all attributes, A_i , and nodes, N_k . Of course, if all information is missing, 4-1 equals 0. In general, function 4-1 will weight more heavily those attributes whose values are more frequently observed (i.e., with greater salience). Throughout chapters 5 and 6, references to category utility will refer to function 4-1. Furthermore, this function is assumed in the version of COBWEB tested in these chapters.

4.6 Learning Superordinate Concepts

COBWEB tends to converge on classification trees where the first level (the root is the 'zeroth' level) is the optimal partition of the observed objects. For notational convenience, as well as anticipating later discussion, the optimal level will be called the *basic* level since it maximizes category utility (see section 3.3.3). Subordinate levels are generated by recursive calls, and are basic levels within their respective subtrees. What remains unspecified is a mechanism whereby *superordinate* nodes are formed. For example, nodes corresponding to *dog*, *cat*, *bird*, *lizard*, etc. might be the basic level in a classification tree over animals, yet a superordinate node representing *mammals* may still be useful.

From a computational standpoint, superordinate categories are intermediaries that are useful for classification based on incomplete information. Consider the case where a furry thing is seen. It is useful to conclude that it is a mammal, from which many other properties can be inferred. In some cases, the use of a salience weight, as described in the previous section, may account for 'superordinate' nodes. For example, because having-hair is such an easily perceivable property, 4-1 will reward a class that covers animals with this property. However, this section presents an alternative mechanism, whereby superordinate nodes are created and accessed in a different manner than are basic level nodes. In particular, a classification tree is constructed so that entry at the basic level is 'hard wired'; superordinate nodes

are accessed only after recognition with respect to the basic level. This is *not* intended as a model of human classification, nor is it an extension to COBWEB that is used in the remainder of the dissertation. Rather, this extension serves to introduce a number of issues and mechanisms important in chapter 7's development of a psychological model of basic level effects. This is the most convenient point at which to introduce some of these issues. However, this section is tangential to the main line of discussion and can be skipped without significantly hindering later understanding of the dissertation.

Given a desire to classify partially described objects, the quality of superordinate concepts can be formalized as a measure of the extent to which individual attribute values predict concepts, thus allowing even partial object descriptions to discriminate concepts. For a concept, N_k , this measure can be formalized as

$$\sum_i \sum_j P(A_i = V_{ij})P(N_k|A_i = V_{ij})^2 \quad (4-2)$$

which can be rewritten as

$$\sum_i \sum_j P(N_k)P(N_k|A_i = V_{ij})P(A_i = V_{ij}|N_k). \quad (4-3)$$

This function determines superordinate concept quality. When averaged over all concepts of a level, it measures partition quality. Like category utility, it is a weighted collocation measure (see 3-4). However, the superordinate measure is biased towards partitions of fewer concepts, and thus acts to compress a basic level partition.³⁰

Two assumptions relegate superordinate concepts to roles secondary to those of the basic level:

- Superordinate concepts may only be modified when the basic level is 'stable'.

³⁰ Superordinate nodes have been implemented in the context of COBWEB's first four operators, but not node promotion.

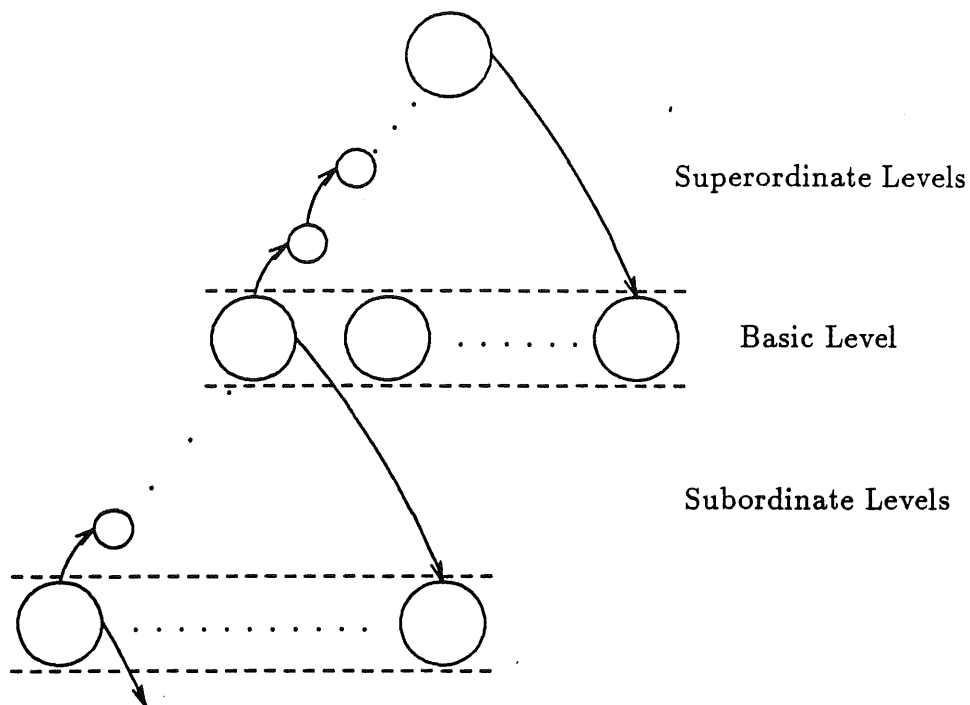


Figure 33

Outline of the object incorporation process in COBWEB.

- Modifications at superordinate levels do not effect the basic level, but basic level changes are made irrespective of their impact on superordinate structure.

Superordinate concepts can only be modified in cases where no new concepts are introduced at the basic level through merging, splitting, or creation for a new object. However, in all cases, the counts of superordinate nodes are updated.

Abstractly, an object is incorporated at the basic level followed by a descent to subordinate levels (via recursive call) and an ascent via parent links to superordinate levels. Controlling ascent requires that nodes be marked as they are visited. This eliminates the possibility that a node's counts are updated more than once. The path followed during incorporation is outlined in Figure 33. A node directly indexes its basic level descendents, bypassing intermediate (superordinate nodes). In addition, each node indexes its immediate children, some of which may

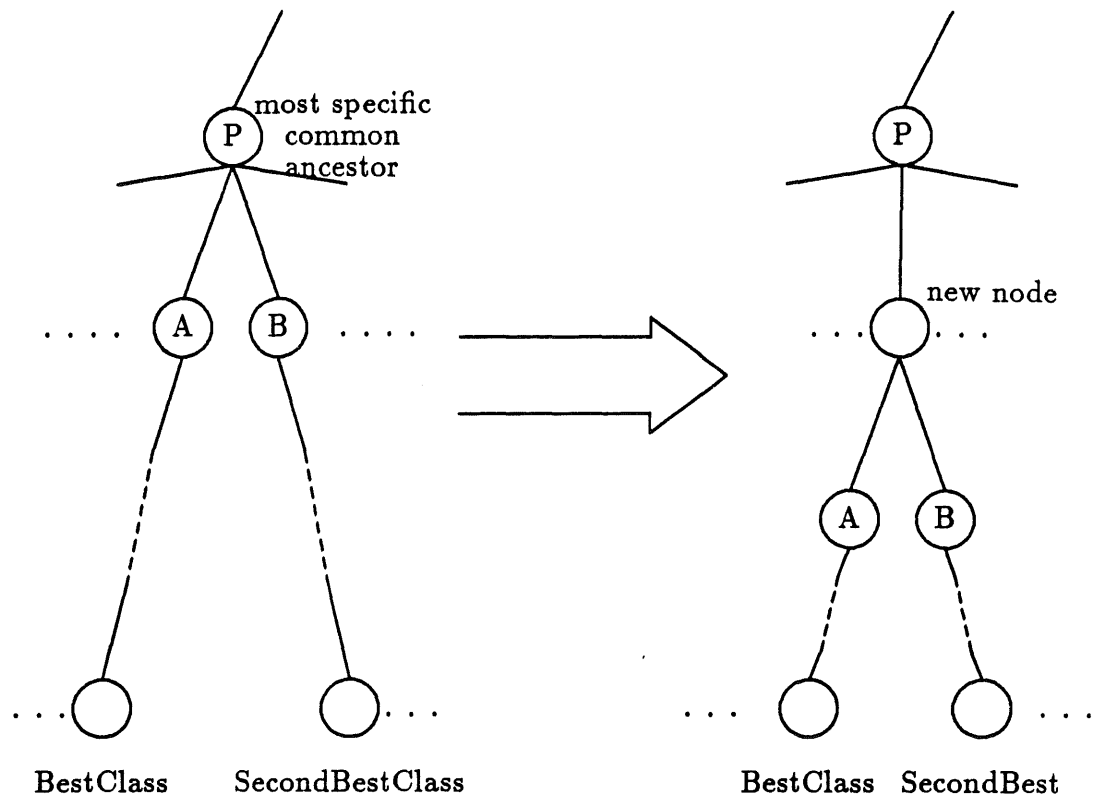


Figure 34

Schematic of superordinate node merging

be basic level descendents as well. In summary, two sets of downward pointing links (children and basic level) are required, as well as parent links.³¹

Creation of nodes at superordinate levels is entirely through merging nodes of lower levels. Destruction of nodes is through splitting. Merging and splitting of superordinate nodes, like their basic level counterparts, are restricted to nodes identified as good hosts for a new object, thus keeping computation cheap. Merging superordinate nodes is considered after determining the two best basic level hosts for an object. The most specific common ancestor of the basic nodes is determined,

³¹ These mechanisms are required if the basic level is 'hard-wired' to be the first level at which classification occurs. However, this is *not* an accurate characterization of human classification (see section 3.3.2). A psychological model of basic level access is presented in chapter 7. Superordinate node formation in the context of the model is given in chapter 8 and significantly simplifies the mechanisms currently being discussed.

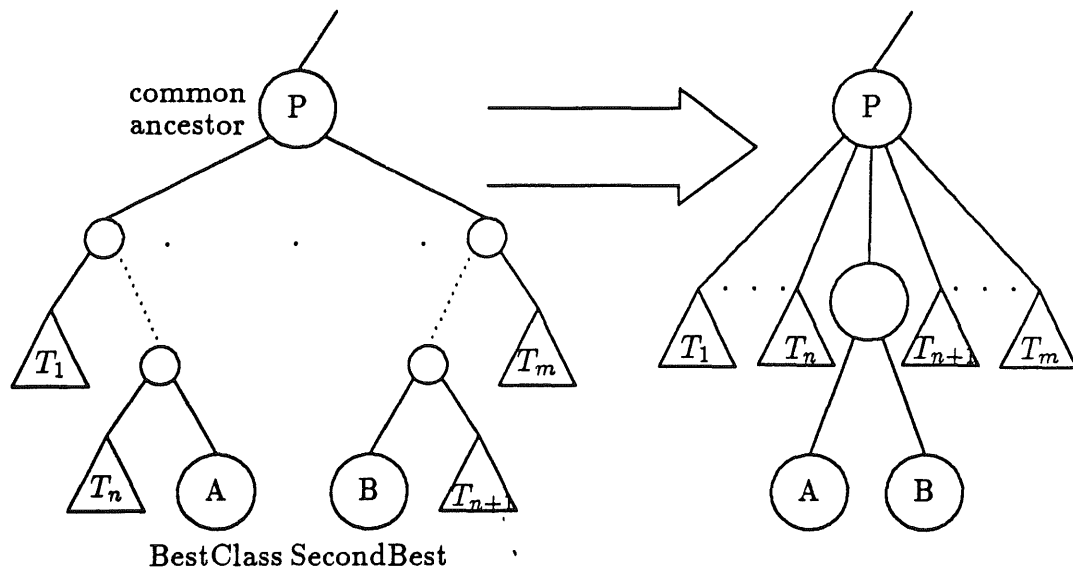


Figure 35

Basic level node merging with superordinate concepts

and the two ancestors of the basic level hosts that are children of this common ancestor are merged and evaluated. This process is shown in Figure 34.

A node is considered an ancestor of itself, so the most specific common ancestor of two basic level nodes may be their parent. Second, it is guaranteed that the most specific common ancestor of two nodes can be no more general than the first marked node encountered. If node merging does not improve the quality of a level according to function 4-1, the result of splitting the ancestor of the best basic level host is evaluated (unless this ancestor is the basic level host itself).

In addition to merging and splitting superordinate nodes, merging (not splitting) of basic level nodes may have significant impact on superordinate structure. When two basic level nodes with distinct parents are merged, neither of the old parents (and nonshared ancestors generally) can be in a parent (ancestor) relation with the new merged node. COBWEB's solution makes the merged node a child of the most specific common ancestor (prior to merging) of the two nodes that were merged. The process is shown in Figure 35.

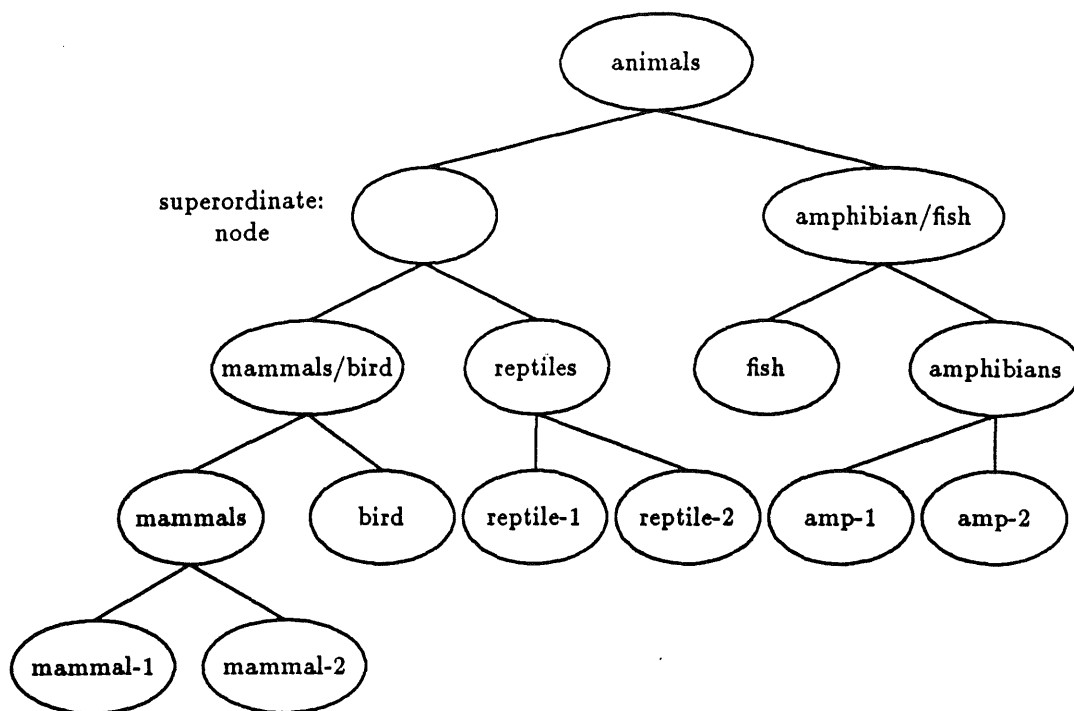


Figure 36

A classification tree with superordinate concept.

Given the data of Table 5, this strategy yields the tree of Figure 36.³² This extension is not assumed in future discussions of COBWEB. However, it is interesting because the control issues that it addresses are similar to those used by a psychological model of basic level effects that is described in chapters 7 and 8. However, rather than hard-coded access to the basic level, the model employs an evidence combination procedure that tends to access the basic level.

4.7 Comparison with Other Systems

According to the taxonomy of chapter 2, COBWEB is a hierarchical conceptual clustering method. It is not strictly agglomerative or divisive, but node

³² A tree of this form was also formed using 4-1. Over several iterations of the data, 'Fertilization' was always observed, while all other attributes were observed only 40% of the time.

merging and splitting allow both types of tree formation. These dimensions distinguish COBWEB from other concept formation systems in several important respects. The goal of this section is to flesh out differences with respect to some selected methods. Discussion focuses on object representation, concept representation, concept quality, hierarchy representation, and control structure.

4.7.1 Object Representation

COBWEB represents objects in terms of nominal attributes only. Recent versions of UNIMEM [LEBO82] allow real-valued attributes. Chapter 9 considers extensions to COBWEB (some underway) that allow hierarchical and continuously-valued attributes. CLUSTER/2 [MIC83A] allows nominal, integer, and *hierarchical* attributes. Hierarchical attributes have domains that are hierarchically decomposed by a 'generalization' hierarchy. For example, color would be a hierarchical attribute if its values were grouped into a hierarchy that included 'red' → 'hot' color → any color. More generally, COBWEB does not operate on structured object descriptions as do CLUSTER/S [STEP84, STEP86] and RESEARCHER [LEB83A].

4.7.2 Concept Representation

COBWEB's concepts differ from those of earlier systems. For example, CLUSTER/2's [MIC83A] concepts are a logical conjunction of (internal) disjuncts. COBWEB's probabilistic representation generalizes CLUSTER/2's representation (i.e., for nominal attributes), since there is a simple mapping from probabilistic to conjunctive concepts. UNIMEM's [LEBO82, LEBO86] concept representation is simpler still; it does not allow internal disjunction. While UNIMEM associates counts with attribute values, they are simply used to determine if an attribute value should be dropped. Only one value of an attribute may be explicitly represented in a concept, or the attribute must be dropped.

COBWEB's representation is most similar to Kolodner's CYRUS [KOL83A]. CYRUS uses counts to compute attribute-value probabilities. In addition, CYRUS identifies certain values as *normative*. These values can be predicted of class members with a high probability. Kolodner treats a value as normative if it is true of at least 67% of the class members. Like CYRUS, COBWEB identifies normative values, but discussion of this ability is deferred until chapter 5. However, while CYRUS does compute probabilities, its use of these numbers is analogous to UNIMEM's use of integer counts. CYRUS only uses probabilities to determine when values should be 'dropped' as norms, whereas COBWEB uses probabilities directly for purposes of object incorporation.

4.7.3 Concept Quality

COBWEB seeks high quality classes and uses an explicit and continuous evaluation function – category utility – to do so. On the other hand, CYRUS and UNIMEM use a binary-valued ('good' or 'bad') evaluation function to determine whether categories should be kept or abandoned. A concept is dropped if it has less than a user-specified number of necessary [LEBO82, LEBO86] or normative [KOL83A, KOL83B] values. The rationale for this rule is that concepts with too few *predictable* values are not useful for inference. Additionally, a concept (i.e., node) can be dropped if all arcs pointing at it are dropped. Recalling chapter 2, each arc is labeled by an attribute value that is *predictive* of the concept. In GBM, a value ceases to be predictive of any node if the number of arcs labeled by that value exceeds a specified threshold. By indexing 'too many' nodes, the value is not predictive of any of them.

As in COBWEB, concept quality in UNIMEM and CYRUS is a measure of attribute-value predictability and predictiveness. However, in UNIMEM and CYRUS it is unclear as to what (if any) function of predictability and predictiveness is being computed. Both systems rely heavily on user-defined thresholds.

'Predictiveness' and 'predictability' appear to be independent of the number of defining attributes, the number of nodes at a level, and the number of instances classified under a node. For example, in GBM it appears that 'green' may be equally predictive of a node containing 5 green objects and 100 green objects.

The use of probabilistic versus logical concepts distinguishes COBWEB from CLUSTER/2. Moreover, COBWEB strives for classes with concepts that facilitate prediction, while CLUSTER/2 desires 'understandable' concepts. However, these representations and performance objectives need not be incompatible [CHEE85, REND86]. Generally, category utility represents a tradeoff between the predictability of attribute values (operationalized as $P(A_i = V_{ij} | N_k)$) and the predictiveness of values (i.e., $P(N_k | A_i = V_{ij})$). An appropriate tradeoff of predictability and predictiveness is necessary in classification structures useful for inference – predictive values combine to direct the classification of partially described objects. Once classified, predictable values can be asserted to complete partial object descriptions. As Medin, Wattenmaker, et. al. [MED86A, MED86B] point out, predictability and predictiveness generalize logical necessity and sufficiency, respectively. Therefore, it is probable that an analogous tradeoff for logical concepts (e.g., Michalski and Stepp's measures of 'simplicity' and 'fit') would result in trees that facilitate inference. Furthermore, chapter 5 argues that using normative values, simple and tight-fitting concepts naturally emerge from COBWEB's search using category utility.

Some may regard the distinction between the understandability versus inference views of concept quality as illusory. However, the distinction is critical in changing the perception of conceptual clustering as useful only for compressing and presenting data in some intuitively good way. Chapter 5 demonstrates that conceptual clustering, guided by appropriate measures of concept quality, is of considerable utility for inference tasks.

4.7.4 Hierarchy Representation

Like other hierarchical methods (e.g., CLUSTER/2 [MIC83A], RUMMAGE [FIS85B], and DISCON [LANG84]), COBWEB groups objects into a strict tree. However, in earlier systems classification proceeds by perfect matching against arc-labeling concepts. In contrast, node-labeling concepts are used for partial matching in COBWEB trees. Partial matching stems from the use of probabilistic concepts and category utility. Of course, probabilistic concepts are not a prerequisite for partial matching. For example, a recent version of UNIMEM [LEBO87] allows exceptions to concept attribute values. While these values were necessary for all class members in earlier versions of UNIMEM (see 4.7.2), the latest version interprets them as 'default' values. A satisfactory partial match occurs when an object shares at least N (a user-defined threshold) of a node's default values. Similarly, a threshold number of normative values must be true for an object to match a concept in CYRUS. The matching functions used by UNIMEM and CYRUS are binary-valued (i.e., 'match' or 'no match') and dependent on constant thresholds. The matching function is not sensitive to the number of objects previously classified, nor the number of attributes used to describe objects. COBWEB's partial matching function is continuously-valued and is sensitive to object and attribute multiplicity. COBWEB selects the concept that maximizes a match with an incoming object.

In contrast to forming strict trees, UNIMEM and CYRUS are clumping methods. These methods form hierarchies in which objects may be classified under multiple nodes. As stated, COBWEB is similar to these methods in that all three use partial matching. Clumping in UNIMEM and CYRUS is a result of independently assessing each class as a possible host for a new object. Kolodner [KOL83A] and Lebowitz [LEBO82] point out that clumping is superior to tree formation, since the former allow multiple interpretations of classified objects. For example, one path may classify an animal as a mammal, while another leads to its identification

as a carnivore. As chapter 3 pointed out, there is good justification for clumping, although it is unclear whether UNIMEM and CYRUS form the 'right' clumps.

4.7.5 Control Structure

COBWEB is an incremental system. Following the general search procedure of chapter 2, COBWEB hill climbs through a space of concept hierarchies, using merging and splitting to insure bidirectional mobility through this space. Through merging, COBWEB shares the bottom-up approach of agglomerative methods. UNIMEM and CYRUS share some of these properties as well. However, unlike UNIMEM and CYRUS, COBWEB can merge arbitrary object classes. UNIMEM and CYRUS are restricted to 'merging' a class with an object description. It is unclear whether this eliminates the possibility of certain object classes, but it seems apparent that this limitation would at least increase the number of observations required to form certain classes.

COBWEB's use of splitting has no analog in UNIMEM, and it is unclear whether there is a similar operator in CYRUS. UNIMEM can only simulate backtracking by dropping subtrees. This is similar to COBWEB's promotion operator. However, node promotion is a more conservative approach to dealing with 'misplaced' nodes. Recall that node promotion can eventually lead to the complete deletion of objects from a tree followed by reclassification. It is unclear under what conditions one technique is more cost effective than the other. The UNIMEM approach requires that all deleted objects be observed again and reclassified. This may be expensive for large subtrees. On the other hand, COBWEB must successively promote and split subtrees. The vigilance required for this may be overkill for small subtrees. An initial hypothesis is that a decision to promote versus delete should be dependent on the size of the subtree in question, but any decision must await further analysis.

Finally, COBWEB reclassifies objects that are removed from a tree, while UNIMEM and CYRUS simply throws them out. Reclassification of objects is appropriate in contexts such as library database organization; it is undesirable to 'forget' library books. Forgetting is more appropriate in contexts such as terrorist news stories. Forgetting versus reclassification is a domain-dependent decision, but a change along this dimension would not effect the basic control structure of COBWEB or UNIMEM in a significant way.

4.7.6 Summary

COBWEB's representations and processing distinguish it from its precursors, CLUSTER/2 and UNIMEM/CYRUS. COBWEB uses probabilistic concepts, a principled matching function, and a search strategy motivated by the constraints of incremental processing. However, COBWEB's development was strongly influenced by past systems. In some aspects it uses abstracted or 'cleaned up' mechanisms from earlier work. This is particularly true of COBWEB's search strategy and concept representation, which find their antecedents in UNIMEM and CYRUS. Along some dimensions (e.g., unconstrained agglomerative behavior), COBWEB extends the control mechanisms of earlier incremental systems. It is not clear whether COBWEB is 'better' than UNIMEM or CYRUS. Rather, this work identifies dimensions along which qualitative comparisons between systems can be made in the first place.

4.8 Chapter Summary

COBWEB is hierarchical and incremental. It can be viewed as hill-climbing through a space of partitions, thus necessarily constraining the search for classification trees to be a hill-climbing procedure as well. Merging and splitting allow bidirectional movement in these spaces. Last, COBWEB uses probabilistic concept representations and a principled evaluation function to guide search.

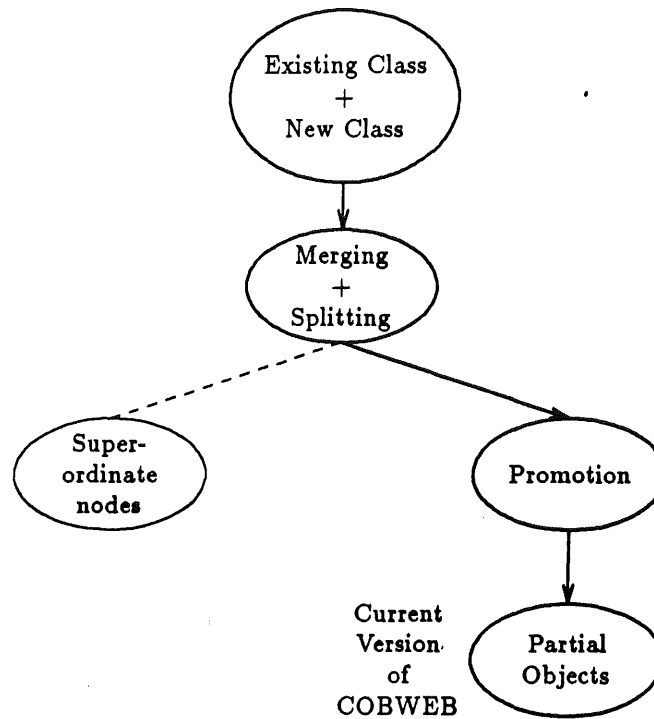


Figure 37

Development history and final version of COBWEB

The following two chapters evaluate COBWEB's output and behavior along a number of dimensions. This evaluation assumes the version of COBWEB depicted at the right leaf of Figure 37. This version includes incorporation into an existing class, new class creation, merging, splitting, node promotion, and the ability to deal with partial object descriptions. Chapter 5 explores the utility of COBWEB classification trees for inference in several domains. Chapter 6 characterizes COBWEB in terms of the cost and quality of learning.

Chapter Acknowledgements

John Gennari's rational reconstruction of a subset of COBWEB's operators led to helpful comments on the exposition of the algorithm.

CHAPTER 5

Classification and Inference

5.1 Chapter Overview

This chapter characterizes COBWEB along two dimensions suggested by the learning model of chapter 1. The model posits three elements surrounding learning: the environment, the knowledge base, and the performance task. This chapter addresses the latter two, while chapter 6 focuses on the impact of incremental presentation of environmental observations.

Section 2 reviews the general form of COBWEB classification trees. They constitute the knowledge base formed by conceptual clustering. Trees from two domains indicate that COBWEB's use of probabilistic concepts does not preclude classes that are characterized by necessary and sufficient properties. Rather, such classes naturally arise from the more general process of looking for classes with predictable and predictive properties.

Section 3 elaborates on predictable and predictive values and defines *normative* values. Normative values can be viewed as *default* values – values that are assumed true unless otherwise stated – with weights that associate a probability of truth with each value. Normative values provide a link between the complete listing of attribute-value probabilities used by COBWEB and symbolic representations.

Section 4 demonstrates the utility of COBWEB classification trees for predicting the value of an object's missing attributes. Unlike a learning from examples system, COBWEB is evaluated by its ability to facilitate prediction of each descriptive attribute, rather than a single teacher-selected 'attribute'. Experiments with soybean and thyroid disease case histories illustrate increases in correct prediction

afforded by COBWEB over an alternative method. The extent of this increase is correlated with a measure *attribute dependence*, which seems to be a reasonable predictor of the 'effort' that is required to 'learn' each attribute.

Section 5 extends the inference experiments of the previous section by demonstrating the cost effectiveness of using normative values during prediction. A prediction in section 4 is made only after classification reaches a leaf. However, norms appear to indicate when classification can cease and predictions can be made with reasonable confidence.

Section 6 compares the performance tasks of learning from examples and conceptual clustering. The former seeks to maximize correct prediction with respect to a single attribute, while the latter seeks to maximize prediction across all attributes. More particularly, COBWEB's prediction abilities are compared with ID3's. ID3 appears to provide a rough upper bound on COBWEB's performance.

Last, experiments of this chapter are concerned with inductive inference - prediction over objects that were not used in tree construction. Section 7 discusses COBWEB's performance as the number of objects used in tree construction increases. Intuitive and empirical arguments suggest that prediction accuracy gradually approaches theoretical upper limits.

5.2 Examples of COBWEB Classification Trees

COBWEB builds classification trees over observed objects. Earlier chapters illustrate the system's behavior and output in the domain of animal descriptions. This section explores COBWEB's output in two other domains. The first demonstrates that probabilistic concepts do not preclude necessary and sufficient conditions. However, relaxing attribute-value necessity and sufficiency within COBWEB's probabilistic framework leads to a definition of *normative* values, which is the focus of section 5.3.

Document Title and Keywords	Attribute - Value Encoding
Cooperative problem solving by like- and mixed-sex teams in a teletypewriter mode with unlimited, self-limited, introduced and anonymous conditions.	Problem-solving = true Words = false Learning = false Incidental-learning = false Sex-differences = true Age-differences = false Computer = true Development = false Recognition = false Visual-stimuli = false Verbal-stimuli = true Perception = false
GROUP PROBLEM SOLVING HUMAN SEX DIFFERENCES VERBAL COMMUNICATION WRITTEN LANGUAGE INTERPERSONAL INTERACTION COMPUTERS MAN MACHINE SYSTEMS	

Table 12

A document high-level and attribute-value description

5.2.1 Example 1: Document Descriptions

Hanson and Bauer [HANS86] use a domain of document descriptions to test their clustering system, WITT. Seven document descriptions (abstract and keywords) were encoded in terms of twelve binary-valued attributes. One document description and its encoding is given in Table 12.

After three iterations through a random ordering of the seven documents, COBWEB formed the tree of Figure 38. Because of space constraints, nodes are labeled only by attribute values common to all node (class) members (i.e., $P(A_i = V_{ij} | C_k) = 1.0$). However, nodes actually contain all observed values and their respective probabilities.

This domain illustrates that COBWEB's processing and probabilistic representations do not preclude classes with perfectly common values. These values can be interpreted in a number of ways. For example, each value with probability 1.0 can be regarded as necessary for class membership. Brachman [BRAC85] calls these values *definitional*. Similarly, a value can be sufficient for class membership. For example, a sufficient attribute value for node N1 of Figure 38 is Problem-solving

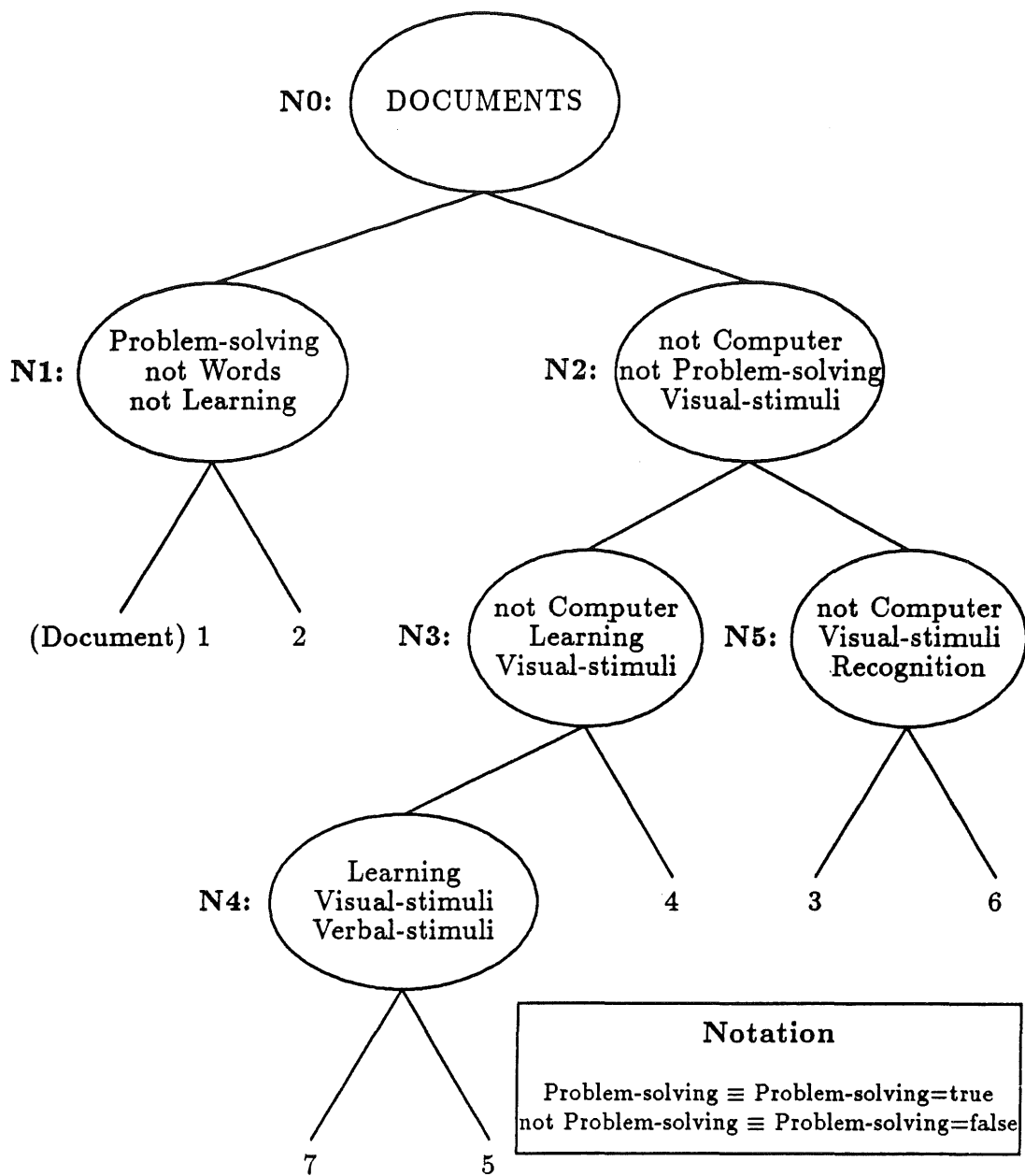


Figure 38

A classification tree over document descriptions

[=true], since $P(N1|\text{Problem-solving} = \text{true}) = 1.0$. Thus, Problem-solving is both a necessary and sufficient value of N1.

Attributes (bills)	Objects(Senators)	
	Goldwater(Rep.-AR)	Cranston(Dem.-CA)
Farm Credit	?	Y
MX Production	Y	N
Budget Cut	Y	N
SDI Cut	N	Y
Contra Aid	?	N
UN Support Cut	Y	?
Relax Gun Control	Y	N
Line-Item Veto	Y	N
Kill School Prayer	Y	Y
Guest Workers	Y	N
Toxics Liability Limit	Y	N
Gramm-Rudman	Y	N
Import Limits	N	N
Farm Bill	Y	Y

Table 13

A sample of senate voting records

Of course, as new objects are added to a class, values that were necessary (definitional) or sufficient may not be so of the altered class. There is no inconsistency in this statement, since the addition of new objects changes a class. Necessity (or sufficiency) applies to the old class, not to the new (different) class. In fact, COBWEB's probabilistic representation allows the evolution of classes with no necessary or sufficient values. This is illustrated in the example below.

5.2.2 Example 2: Congressional Voting Records

A domain that shows the flexibility of probabilistic concepts is congressional voting records.³³ Members of the U.S. Senate were represented in terms of 14 key votes taken in 1985. Key votes were designated by the *Congressional Quarterly* [CONG85] and ranged from domestic issues like emergency farm credits, gun control, and school prayer, to foreign affairs issues such as Nicaraguan 'Contra' aid. Party

³³ This domain was inspired by a similar one used by Lebowitz [LEBO85].

	N_1 ('conservative')	N_{79} ('liberal')
Typical values	Toxic-Waste - yes (0.81)	Toxic-Waste - no (0.88)
$P(\text{value} \text{node})$	Budget Cuts - yes (0.81)	Budget Cuts - no (0.90)
	SDI reduction - no (0.93)	SDI reduction - yes (0.83)
	Contra Aid - yes (0.88)	Contra Aid - no (0.83)
	Line-Item Veto - yes (0.91)	Line-Item Veto - no (0.86)

Table 14

Some typical values of 2 clusters from congressional hierarchy

affiliations (Democrat and Republican) were not included in the representation. Each attribute corresponded to one of the 14 votes, with each attribute having two possible values, 'yes' or 'no'. Two 'objects' are shown in Table 13, while the complete data set is given in the appendix to the dissertation. Question marks denote unknown values.

As might be expected, COBWEB formed a classification that roughly groups senators as 'liberals' and 'conservatives' at the top level. Democrats predominantly inhabit the 'liberal' class and republicans dominate the 'conservative' class. Figure 38 shows some of the values and probabilities associated with these nodes.

Lower level nodes serve to further distinguish the groups at the top level. For instance, a 'conservative' subnode contains eight of the ten democrats in the 'conservative' class. This smaller group corresponds to the concept of 'southern democrat' or alternatively 'Jackson democrat',³⁴ which differ from other 'conservatives' by opposing budget cuts (with probability 0.92).

This domain differs from the document domain by its lack of necessary (and/or sufficient) conditions for top-level classes. None of the 'conservative',

³⁴ Seven of eight senators were from southern states. A 'Jackson' democrat is one that votes conservatively on military/foreign policy issues and liberally on social issues.

'liberal', or 'southern democrat' clusters contained any perfectly common values. A relaxation of necessity leads to the important idea of *normativeness*.

5.3 Default and Normative Values

A normative value is one that is true with a reasonable degree of confidence. Normativeness is a probabilistic notion that captures a tendency, rather than a rule, of an object class. Attribute-value normativeness can be naturally defined within the probabilistic representations of COBWEB. However, using logical and pseudo-logical representations, many researchers have employed the analogous idea of a *default* value. This section motivates a definition of normativeness, in part, by generalizing non-probabilistic conceptions of defaults.

5.3.1 Definitional and Default Values

Values that are not necessary (or definitional) of a class may still be regarded as *default* values [BRAC85, ETHE83, REIT80]. In the case where a value, V , is definitional of a class, it can be stated that "If an object is a member of the class then the object has value V ." In the case where V is a default value of a class, it is said that "If an object is a member of the class then the object has value V , *unless shown otherwise*". There are several ways that a default can be overturned (i.e., *shown otherwise*). One way is to simply be told otherwise: "The elephant isn't *gray* (the default), its *pink*." More generally, a default can be vetoed by a proof based on known facts.

Consider a simplified structuring of the congressional voting records described earlier. Assume the deductive rules,

$$\forall x \text{ Senator}(x)$$

$$\text{Senator}(x) \wedge \text{Contra-aid}(x, \text{yes}) \wedge \text{SDI-reduce}(x, \text{no}) \rightarrow \text{Conservative}(x),$$

$$\text{Conservative}(x) \wedge \text{Line-item-veto}(x, \text{no}) \rightarrow \text{Southern-democrat}(x), \text{ and}$$

$$\text{Southern-democrat}(x) \wedge \text{State}(x, \text{alabama}) \rightarrow \text{Denton}(x).$$

Furthermore, assume the default rules,

$\text{Conservative}(x) \rightarrow \text{Budget-cuts}(x, \text{yes/default})$ and
 $\text{Southern-democrat}(x) \rightarrow \text{Budget-cuts}(x, \text{no/default})$.

Last, a final deductive rule is

$\text{Denton}(x) \rightarrow \text{Budget-cuts}(x, \text{yes})$.

Knowing $\text{Conservative}(x)$ suggests that x voted 'yes' to budget cuts. However, if deductive rules determine $\text{Southern-democrat}(x)$, the earlier default can be replaced by the guess that x voted 'no' to budget cuts. In turn, this can be overturned should it be found that $\text{Denton}(x)$.

Deductive and default rules of this form can be compiled into a decision tree-type structure. The rules above imply the partial decision tree of Figure 39. The tree includes four classes: Senators, Conservatives, Southern-democrats, and the singleton class, Denton. Conservatives and Southern-democrats have associated defaults, while Denton has a definitional value. Classification via the decision tree is a simple proof procedure. For example, if x is a conservative then x voted yes to budget cuts, unless x can be shown to be a Southern-democrat, in which case x voted no, unless x can be shown to be Denton, in which case x voted yes.

Figure 39 demonstrates how defaults can change and even cycle as more knowledge is brought to bear. One advantage of defaults is that if limited knowledge makes deduction impossible, an inductive guess still offers hope of guessing an observation's missing properties. A second advantage of default values is they can save space. Consider the additional classes of Figure 39, Reagan-supporters and the singletons, Laxalt and Hollings. There are no explicit defaults (Budget-cuts) associated with these classes. By convention, they *inherit* (rather than replace) the

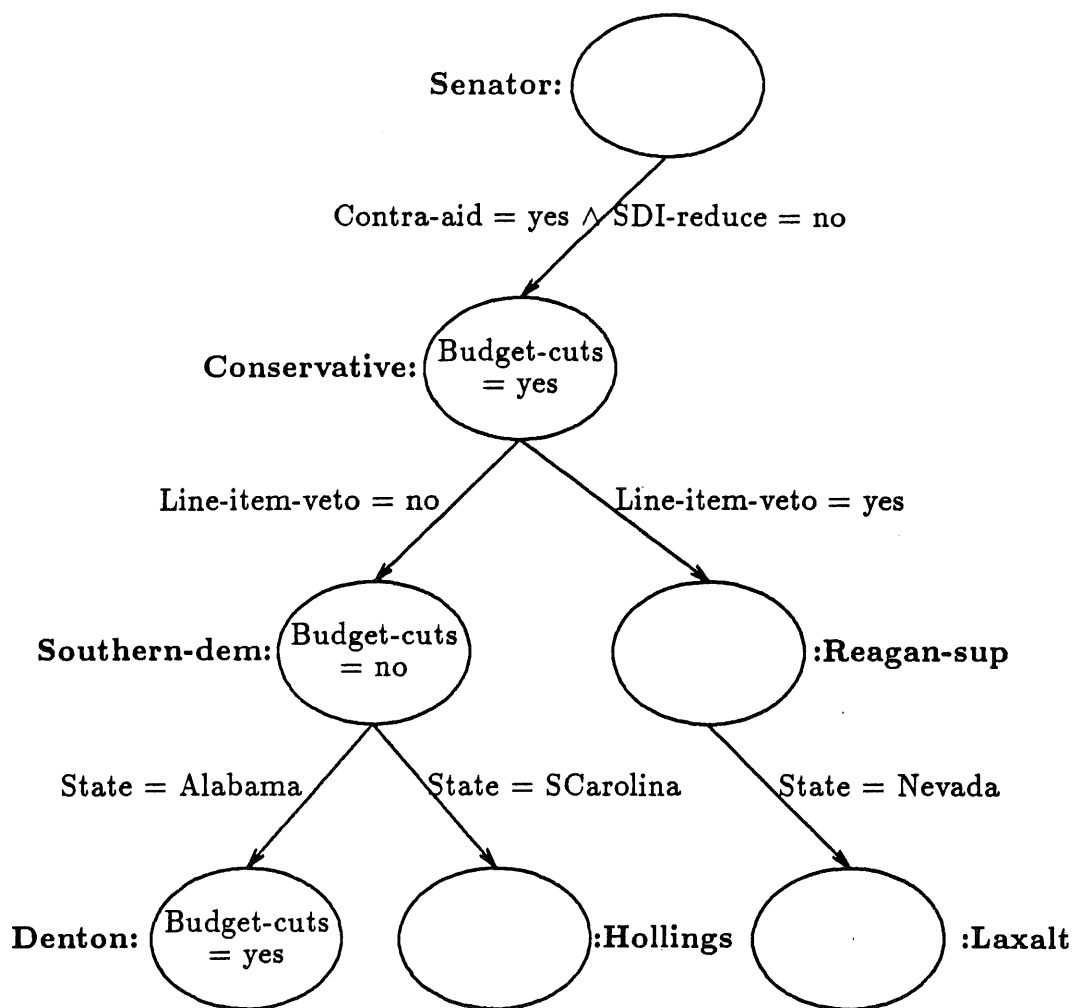


Figure 39

Nested classes with defaults.

defaults of their ancestors. Hollings voted 'no' to budget cuts and Laxalt voted 'yes'.

Discussion has stipulated how defaults are used, once selected. However, there has been no mention of how defaults are selected. Intuitively, defaults might correspond to usually occurring or *typical* values. However, the literature on default values makes almost no mention of prescriptive means for assigning default values. Brachman [BRAC85, p 84] states "In the manner used in frame notations, however,

a [default] value could be violated in every single case!" Lacking guidelines for default assignment there is, not coincidentally, little (if any) discussion on default value maintenance in the context of learning. Brachman also points out that frame-based systems typically make no distinction between default and definitional values.

COBWEB is intimately concerned with learning and dealing with changing class structure. *Normative* values generalize defaults by attaching probabilistic qualifiers to values of a class. Probabilities encourage an interpretation of normative (default) values as typical ones. In turn, this provides a prescriptive view of normativeness that is not thwarted by changing class structure.

5.3.2 Normative Values

Kolodner [KOL83A, KOL83B] defines a normative value as one that is true of at least $2/3$ (67%) of a class's members. By Kolodner's definition, all values in Table 14 are norms of their respective classes. In general, normative values can be treated as defaults, but probabilities indicate the degree to which exceptions exist. Furthermore, a natural distinction between definitional and default values exists. The former are true of class members with probability 1.0, while the latter have a probability of less than 1.0.

Constant threshold strategies, such as Kolodner's, specify when a property is reasonably true. However, this technique does not address the issue of when it is reasonable to generate a guess. The distinction between these two points is subtle. Certainly, if a property is 100% assured (definitional), it is reasonable to generate a prediction. However, if one value is true 67% of the time, is it reasonable to continue classification based on further information (if available) or to generate a prediction?

Quinlan's learning from examples system, ID3, addresses this problem. ID3 is concerned with prediction of a single 'attribute': a teacher's definition of class membership. Recalling chapter 2, ID3 stops tree building when class membership

is independent (according to a chi-squared measure) of all other attributes. That is, no attribute aids in predicting teacher-defined class membership and so further decomposition is not worth the effort. The case where teacher-defined class membership ($= C_k$) is definitional of a node is a special case of this independence condition; all objects at the node are members of C_k , regardless of their value along any other attribute. In general, attributes with one value of high probability (e.g., 0.67) tend to more closely approximate independence from other attributes. A view of normativeness based on independence promises to generalize constant threshold strategies and is not dependent on an arbitrary cutoff.

5.3.2.1 A Definition of Norms

In COBWEB, a value can only be normative at a node where the attribute approximates conditional independence from other attributes. A determination of (approximate) independence is not made directly, but indirectly by looking for nodes that maximize a function of value predictability and predictiveness. In particular, a necessary, but not sufficient, condition for a value, V_{ij} , to be a norm at a node, N_k , is that the value maximize

$$P(A_i = V_{ij}|N_k)P(N_k|A_i = V_{ij}).$$

Recalling chapter 3, this product is the *collocation* of V_{ij} with respect to N_k . An example from 3.3.3 illustrates that in a animal taxonomy containing {animal, bird, robin}, the collocation for 'Mode-of-transport = flies' would be maximized at the 'bird' node since the predictability and predictiveness of this value are both high for this class.

An important property of collocation is that if an attribute, A_i , is rendered conditionally independent of other attributes at a node, then the collocation of each of the attribute's values will be maximized at that node. For example, consider

a subtree rooted at N_k with children, C_1 through C_n . If A_i is independent of membership in C_l then by the definition of independence,

$$P(A_i = V_{ij}|C_l) = P(A_i = V_{ij}|N_k),$$

for all children of N_k . In addition, recall (from 3.3.3) that for a node, N_k ,

$$P(N_k|A_i = V_{ij}) \geq P(C_l|A_i = V_{ij})$$

for all children, C_l . Thus, given A_i is independent of membership in C_l ,

$$P(A_i = V_{ij}|N_k)P(N_k|A_i = V_{ij}) \geq P(A_i = V_{ij}|C_1)P(C_1|A_i = V_{ij}).$$

That is, N_k maximizes the collocation of V_{ij} with respect to its children. In fact, under this assumption, N_k maximizes the collocation of V_{ij} with respect to all of its descendants. Because collocation for an attribute's values are maximized at nodes that render the attribute independent of lower level classes, norms will typically be identified at nodes where it is reasonable to generate a prediction; going lower in the tree will not improve prediction accuracy. Importantly, the relation between collocation and independence is one-way: attribute independence at a node insures that the collocation of the attribute's values will be maximized at the node with respect to its descendants, but not vice versa. However, an attribute with collocation-maximizing value(s) may *approximate* independence from lower level classes.

Despite the relation between collocation and independence, perfect attribute independence is rarely attained. However, if the probability of one value of an attribute is relatively high, the attribute may approximate independence from lower level classes. In particular, a node N_k with high $P(A_i = V_{ij}|N_k)$, will tend to have less variance in the value of A_i over the children of N_k . The less variance over

these children, the closer A_i will approximate independence from other attributes. Of course, $P(A_i = V_{ij}|N_k)$ is maximized at nodes where V_{ij} is definitional ($= 1.0$). This is guaranteed (but not exclusively so) to occur at leaves (singleton classes). Favoritism towards specific nodes is tempered by consideration of $P(N_k|A_i = V_{ij})$ as this probability tends to be higher for more general classes. The tendency of $P(N_k|A_i = V_{ij})$ to increase with the generality of N_k causes collocation to be maximized before absolute conditional independence is achieved.

To summarize, collocation for each of an attribute's values will be maximized at a node that renders the attribute conditionally independent of lower level nodes. This will insure that norms indicate when prediction accuracy will not improve by deeper classification; this was an important lesson learned from ID3. Second, a value that is highly probable will mean less variance over all values of the corresponding attribute and thus the attribute may approximate independence from lower level nodes. In general, at most one value of an attribute can be normative of a class: the value must maximize collocation at the class *and* the value must be the most probable of all values. Together, these conditions generalize constant threshold strategies of selecting norms as done in CYRUS.

5.3.2.2 Examples of Norms

Consider how norms are determined for the tree formed in the congressional domain. In particular, consider the lineage in Figure 40. With each of the six nodes are the collocation scores for 'Budget-cuts = yes'. The collocation of this value is maximized at the 'Conservative' (N1) node. While collocations are only shown for one ancestral line, in fact, the collocation of 'Budget-cuts = yes' at the 'Conservative' node is greater than at *any* of its descendents (and not just N2, N18, N24, and N29). In addition, $P(\text{Budget-cuts} = \text{yes}|N1) = 0.81 \geq P(\text{Budget-cuts} = \text{no}|N1) = 0.29$. Thus, 'Budget-cuts = yes' is regarded as a norm of the 'Conservative' node.

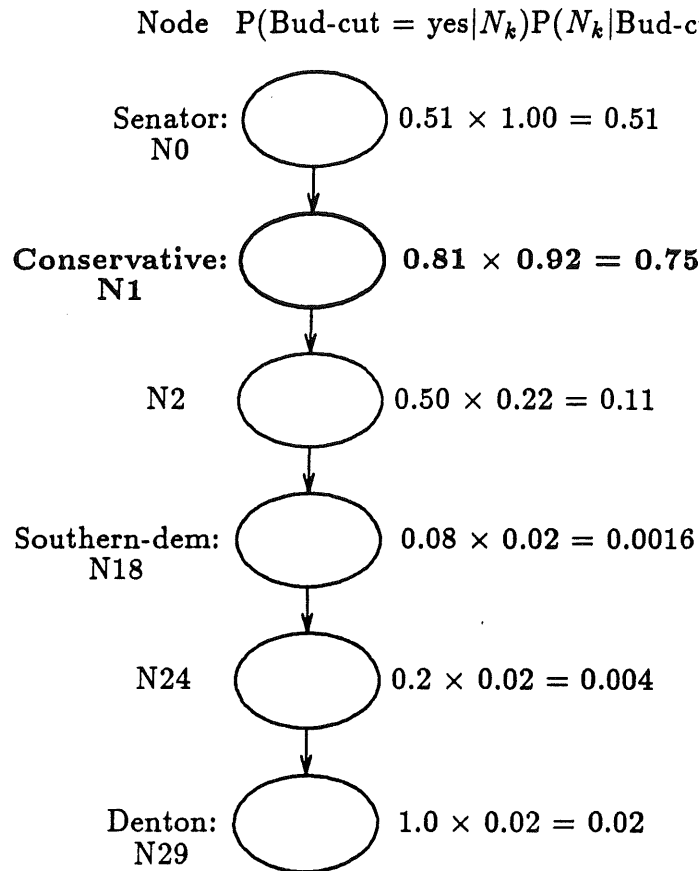


Figure 40

Determining a node where Budget-cuts = 'yes' is normative

Figure 41 shows similar computations for 'Budget-cuts = no'. For the ancestral line used in the first (i.e., 'Budget-cuts = yes') example, the collocation for 'Budget-cuts = no' is greatest at the root. However, the root does not maximize collocation with respect to all of its descendents. As Figure 41 shows (and Table 14 alluded to), collocation for 'Budget-cuts = no' is maximized at the 'Liberal' node and it otherwise satisfies the conditions for normativeness.

5.3.2.3 Subnorms

The norms of the last two figures are computed from the predictability of an attribute value at a node and the value's predictiveness towards the node in

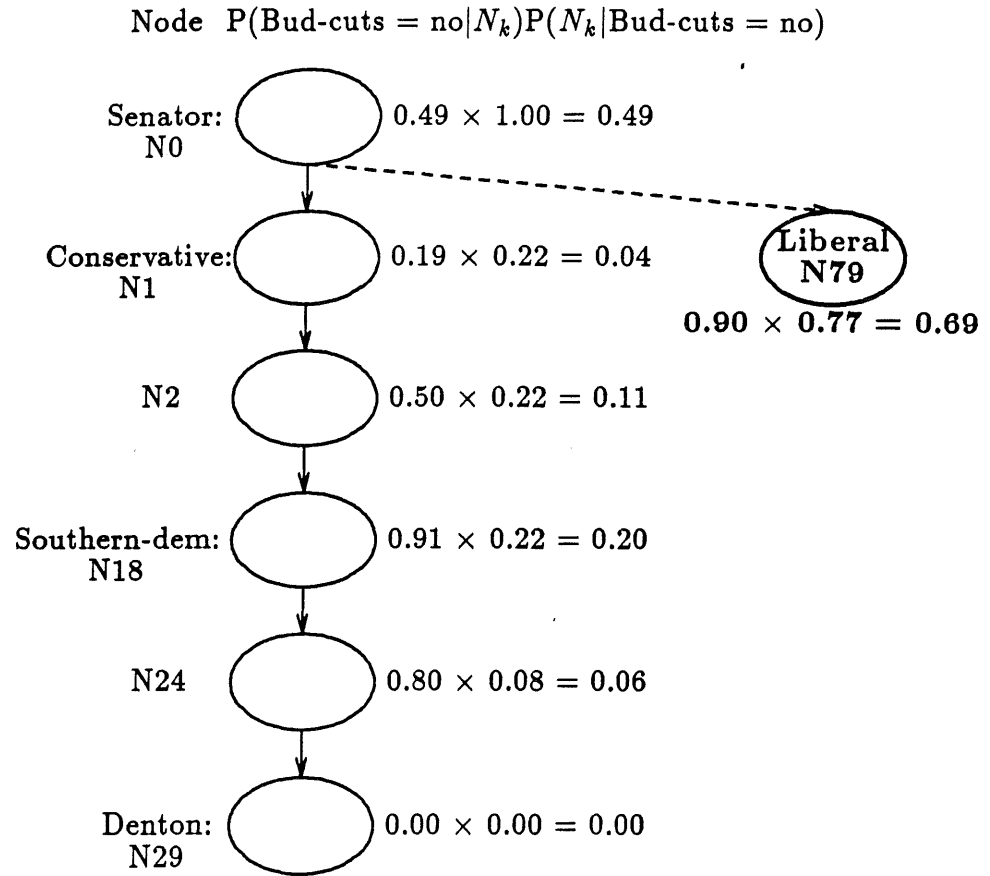


Figure 41

Determining a node where Budget-cuts = 'no' is normative

question; predictiveness has been from the root of the tree to the node. For example, the predictiveness of 'Budget-cuts = no' for node N2 in Figure 41 is $P(N2|\text{Bud-cuts=no}) = 0.22$. This value is based on root level statistics. It is also possible to compute the predictiveness of 'Budget-cuts = no' towards a N2 from node N1. This predictiveness score is *conditioned on* prior classification with respect to node N1. That is, $P(N2|\text{Bud-cuts=no}, N1) = 1.0$. Because the norms of the previous examples were based on statistics accumulated over all observed objects and stored at the root, they are called *unconditioned* norms;

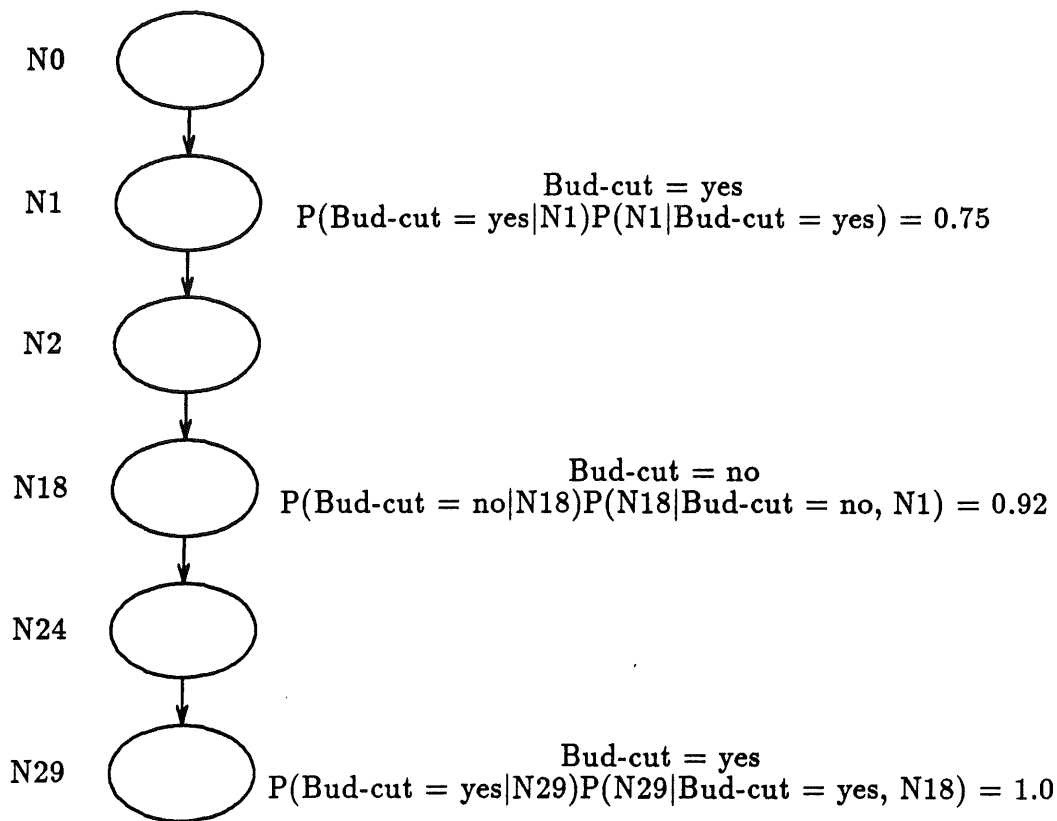


Figure 42

Determining subnorms for Budget-cuts

the computation of predictiveness is not conditioned on nodes that are subordinate to the root.

Once normative values have been defined at a node, it is possible to define *subnorms*. Subnorms are values that contradict and supercede normative values at higher level nodes. They are similar in form and intent to the nested default values shown in Figure 39. Using unconditioned norms, subnorms can be identified. Although no unconditioned subnorms were identified in the last two figures, if an unconditioned norm for 'Budget-cuts = no' had been found that was subordinate to node N1 of Figure 41, this would supercede 'Budget-cuts = yes' at N1.

More generally, subnorms are not restricted to unconditioned norms. In particular, *conditioned* norms are identified by assuming that the constituent probabilities of collocation are conditioned on membership in the last class containing a norm along the appropriate attribute. For example, consider Figure 42. 'Budget-cuts = yes' has already been identified as a norm of the 'Conservative' (N1) node. To determine subnorms, collocation (conditioned on N1) for 'Budget-cuts = no' is computed for each descendent of N1. These computations indicate that the 'Southern democrat' (N18) node contains 'Budget-cuts = no' as a subnorm. Similar computations (conditioned on membership in N18) can spot normative values of still lower nodes.

5.3.2.4 *The Cost of Identifying Norms*

Identification of normative values in COBWEB appears, in the worst case, to require greater computation than using a constant threshold strategy. The possibility of greater complexity arises because a maximal collocation value must be determined with respect to an ancestral line. In the worst case, this might involve checking every node in an ancestral line, as opposed to a constant threshold strategy, which requires only descending until a value's probability surpasses a threshold. Each of the three previous figures demonstrate two frequently observed properties that are important in this regard. One is that the first local maximum of the collocation score (as one descends) tends to be the global maximum. For example, the collocation for 'Budget-cuts = yes' is 0.75 at node N1. This is the first local maximum and the global maximum as well. This observation offers hope that norms can be efficiently determined. However, collocation does not necessarily decrease in an orderly manner as one descends the tree. There may be many local maxima. It does not appear possible to prove the first local maximum is necessarily the global maximum.

N_1 ('conservative')	N_{79} ('liberal')
$P(A_i = V_{ij} N_1), P(N_1 A_i = V_{ij})$	$P(A_i = V_{ij} N_{79}), P(N_{79} A_i = V_{ij})$
Toxic=Waste = yes (0.81,0.90)	Toxic=Waste = no (0.88,0.78)
Budget Cuts = yes (0.81,0.92)	Budget Cuts = no (0.90,0.78)
SDI reduction = no (0.93,0.88)	SDI reduction = yes (0.83,0.90)
Contra Aid = yes (0.88,0.88)	Contra Aid = no (0.83,0.83)
Line=Item Veto = yes (0.91,0.90)	Line=Item Veto = no (0.86,0.88)
MX Production = yes (0.90,0.95)	MX Production = no (0.93,0.87)
Guest Workers = yes (0.77,0.83)	Guest Workers = no (0.78,0.71)
	Gramm=Rudman = no (0.54,0.92)
Farm Bill = yes (0.81,0.82)	Farm Bill = no (0.75,0.73)

Table 15

Normative values for congressional classes

Despite the difficulty with proof, experimentation indicates that the first collocation maximum tends to be the global one. Intuitively, decreasing value predictiveness encourages this. Therefore, once there is a decrease in collocation, an experimentally validated rule is that the prior node maximizes collocation. Thus, in general a one node 'look-ahead' strategy appears adequate for determining node norms. This strategy is comparable in cost to norm identification using a constant threshold strategy. Chapter 8 returns to the problem of a cost-effective identification of norms since COBWEB/2 needs to dynamically identify norms. However, dynamic identification of normative values is not important from the standpoint of incorporating objects in COBWEB. Rather, the importance of individual values is implicit in the category utility calculation. Instead, from COBWEB's perspective, normative values provide a link between probabilistic and symbolic representations that can be exploited *after* learning has occurred.

5.3.3 Normative Values and Symbolic Descriptions

The use of probabilistic (versus logical) concepts distinguish COBWEB from the work of Michalski and Stepp [MIC83A]. However, these representations need not be incompatible [CHEE85, REND86]. The gap between probabilistic and symbolic representations is bridged by normative values. Consider COBWEB's selection of normative values in Hanson and Bauer's document domain. These correspond to the necessary and sufficient values listed with the classification tree nodes of Figure 37. In Michalski and Stepp's terminology, necessary and sufficient values represent concepts that are simple and tightly fit the data.

In cases where there are no (simply stated) necessary and sufficient conditions, normative conditions may still be used to symbolically describe classes. Consider the complete set of normative values selected for nodes N_1 ('Conservative') and N_{79} ('Liberal') of the congressional domain and shown in Table 15. Members of these classes tended to be at opposite poles on many issues. A symptom of this is that for every norm of the 'Conservative' class there is a corresponding norm of the 'Liberal' class with an opposite vote on the same issue. Conversely, of the 'Liberal' norms, only 'Gramm-Rudman = no' has no corresponding norm in the 'Conservative' class. In general, siblings of a COBWEB classification tree need not have such a preponderance of norms defined along the same attributes.

Collectively, almost any three norms from the 'Liberal' or 'Conservative' class can be used to (perfectly) distinguish members of that class. For example, a rule of the form

If any two of the values, {Toxic-waste = yes, Budget-cuts = yes, SDI-reduction = no}, is true of an senator, then he/she is a 'Conservative',

can be used to distinguish 'Conservatives' from 'Liberals'. This is an example of a *polymorphous* concept [HANS86]. In general, a list of normative values is regarded

as a class prototype and any object with a specified number of these values can be regarded as a class member. In the example above the specified number is two. More generally, this threshold may be constant or variable (e.g., a majority). In the congressional domain, two normative values are sufficient to distinguish classes, but this need not be the case in other domains.

A large collection of normative values tends to be an emergent property of using category utility to guide class formation. Specifically, the numerator of category utility (3-7) of a partition can be expressed as

$$\sum_k \sum_i \sum_j [P(A_i = V_{ij})P(N_k|A_i = V_{ij})P(A_i = V_{ij}|N_k)] \\ - \sum_i \sum_j P(A_i = V_{ij}) \times 1 \times P(A_i = V_{ij}),$$

where $1 = P(\text{root}|A_i = V_{ij})$ for each V_{ij} . Furthermore, $1 \times P(A_i = V_{ij})$ is the collocation of V_{ij} at the root. Category utility tends to favor classes having values with collocation scores greater than the root's, i.e., $P(N_k|A_i = V_{ij})P(A_i = V_{ij}|N_k) \gg 1 \times P(A_i = V_{ij})$. Intuitively, category utility favors classes with many norms.³⁵

Experimentally, classes with many norms are found to have simple and tight-fitting symbolic descriptions in the sense used by Michalski and Stepp [MIC83B]. That is, polymorphic concepts for each class will require few norms to distinguish classes. Additionally, norms are values that can be predicted with reasonable assurance; many norms offer greater inference possibilities. Thus, the performance objectives of COBWEB (inference) need not oppose an earlier view of conceptual clustering, which favors understandable concept descriptions.

Finally, while probabilistic concepts are typically justified because they generalize logical (typically conjunctive) representations [SMIT81, HANS86], the former

³⁵ Since the basic level is hypothesized to be where category utility is maximized, a characteristic of the basic level may be that it has more normative values. This discussion also relates to Jones' [JONE83] hypothesis that the basic level is where the most collocation maximizing nodes are present. See section 3.3.3 for more details.

seem more suitable for incremental systems (Schlimmer, personal communication). Even heuristic measures applied to logical representations may be computed from summary statistics. However, in the context of nonincremental systems these statistics can be computed as needed and there is no need to make them an explicit part of the concept representation. In incremental systems it is advantageous to maintain summary statistics, thus reducing cost when incorporating new objects. Nonincremental methods tend to compute statistics as necessary, whereas an effective incremental strategy is to generate symbolic descriptions as necessary.

5.3.4 Summary

COBWEB's definition of normative values generalizes the notion of default values by assuming probabilistic qualifiers. This provides a clear prescription of how default values are identified and maintained over a changing classification structure. Normative values indicate tendencies in data. This does not preclude a representation of necessity, but generalizes it. Identification of strong tendencies in data are primarily useful for purposes of inference, a task that is explored in the next section.

5.4 COBWEB Classification Trees and Inference

COBWEB seeks classifications that maximize the information inferrable from category membership. The efficacy of this domain-independent heuristic requires that important properties be dependent on regularities or 'hidden causes' [PEAR85, CHEN85] in the environment, and that these regularities be identified by a conceptual clustering system.

Experiments in two domains indicate the utility of COBWEB's classification trees as predictive models. More generally, this reinforces a view of inference as a byproduct of classification. Last, comparisons with ID3, a program that learns

Attributes	Domains	Example
Time of occurrence	(april, may, june,..., october)	october
Plant stand	(low, normal)	normal
Precipitation	(low, normal, high)	high
Temperature	(low, normal, high)	normal
Occurrence of hail	(yes, no)	no
Severity	(minor,potential,severe)	potential
Leaf condition	(normal, abnormal)	abnormal
.		
.		
.		
Diagnostic Condition	(Stem Canker, ...)	Stem Canker

Table 16

Sample soybean disease cases

from examples, indicate a weakness in COBWEB's reliance on strict trees as a way of structuring observations.

5.4.1 Learning to Diagnose Soybean Disease

The first domain was a set of 47 soybean disease cases taken from [STEP84]. Each case (object) was described along 35 attributes. Four categories of soybean disease were present in the data - Diaporthe Stem Rot, Charcoal Rot, Rhizoctonia Root Rot, Phytophthora Rot. These disease designations were also included in each object description, making a total of 36 attributes. An example case description is given in Table 16.

Soybean cases were presented to COBWEB in order to see whether the resultant classification could be used for effective disease diagnosis. While Diagnostic-condition was included in each object description, it was simply treated as another attribute. In building a classification tree, Diagnostic-condition did not force a classification as in learning from examples. After incorporating every fifth instance, the remaining *unseen* cases were classified (but not incorporated) with respect to the classification tree constructed up until that point. Thus, like studies done by

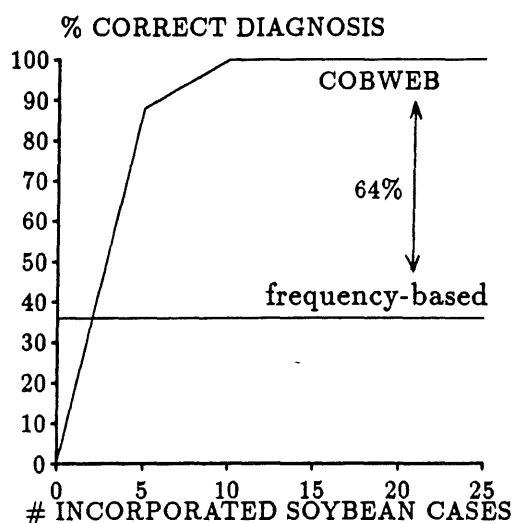


Figure 43

Diagnostic success with soybean cases

Quinlan [QUIN83], the input was implicitly divided into *training* and *test* sets. The goal was to determine if clustering over the training set improves inference over the test set through a process of classification.

Classification was performed in a manner similar to incorporation, except that statistics were not updated, nor were merging, splitting, or class creation performed. That is, a test object was tentatively placed in each class of a set of siblings. The class that maximized the category utility of the resultant partition was selected as the best host. The object was then classified recursively with respect to the best host. Test instances contained no information regarding Diagnostic-condition, but the value of this attribute was inferred through classification. Specifically, classification terminated when the test object was matched against a leaf of the classification tree. The leaf represented the previously observed object that best matched the test object. The Diagnostic-condition of the test object was predicted to be the corresponding condition of the leaf. This procedure is given as one type of *case-based reasoning* by Kolodner [KOL087].

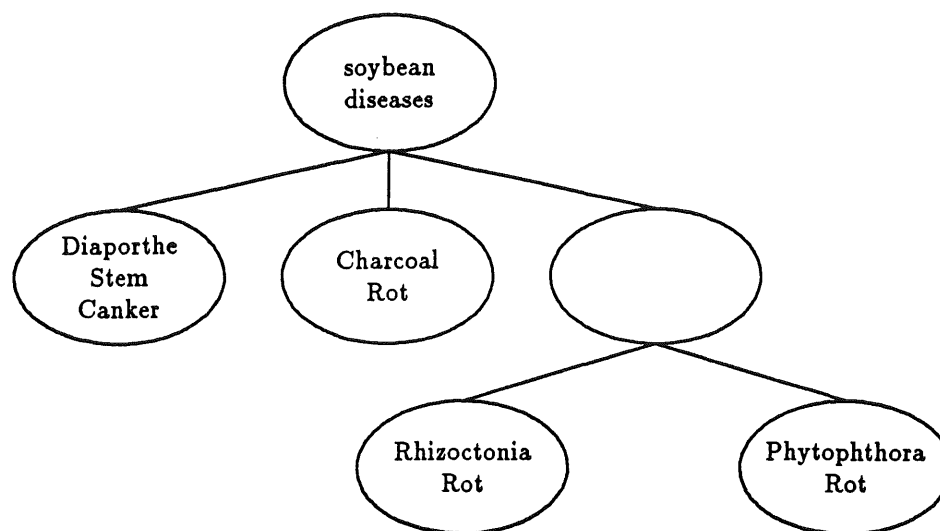


Figure 44

The top levels of a classification tree over soybean disease cases

Figure 43 gives the results of the experiment. The graph shows that after 5 instances the classification could be used to correctly infer Diagnostic-condition (over the remaining 42 unseen cases) 88% of the time. After 10 instances, 100% correct diagnosis was achieved and maintained. To put these results into perspective, Figure 43 also graphs the results obtained by a simpler, but reasonable, inferencing strategy. This 'frequency-based' method dictates that one always guess the most frequently occurring value (Phytophthora Rot) of the unknown Diagnostic-condition attribute. This method gives a 36% correct prediction rate. Thus, the COBWEB classification tree facilitates a 64% increase in correct prediction.

While impressive, these results follow from the great regularity of this domain. Members of the different diagnostic conditions are sufficiently different that having seen one instance of each diagnostic condition insures good prediction along this attribute.³⁶ In fact, when COBWEB was run on the data with no information

³⁶ The rapidity with which diagnostic condition was 'learned' may be surprising to some readers. To bolster claims about the generality of this observation, 10 more experiments were made on randomly generated orderings. On eight of the ten trials, 100% correct diagnosis was obtained by 25 instances, while 96% and 91% accuracy was obtained on the remaining two trials. On six trials 100% correct

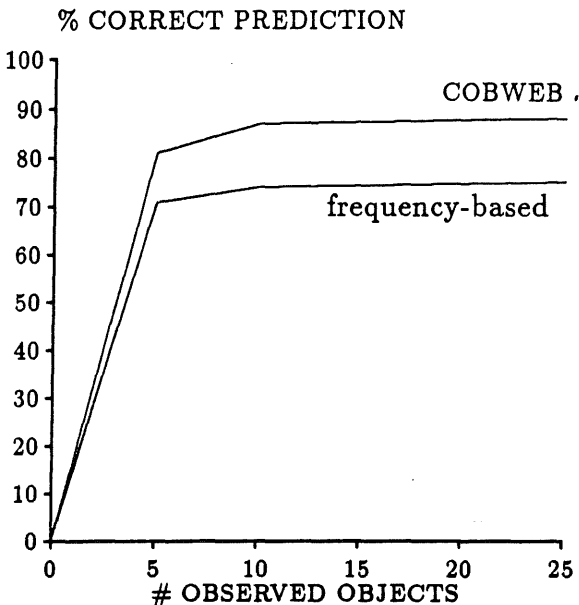


Figure 45

Success at inference averaged over all soybean attributes

of Diagnostic-condition at all, the four classes were 'rediscovered' as nodes of the resultant classification tree (Figure 44).³⁷ This indicates that Diagnostic-condition participates in a network of attribute correlations. Forming classes around these correlations is rewarded by category utility, resulting in classes corresponding to the human-defined diseases.

The success at inferring diagnostic condition implies a relationship between an attribute's dependence on other attributes and the utility of COBWEB classification trees for induction over that attribute. To further characterize this diagnosis was obtained after 9 or fewer trials. One trial only required 7 instances to achieve 100% correct diagnosis.

³⁷ Stepp's CLUSTER system (Stepp, 1984) also rediscovered the disease classes. However, unlike CLUSTER, COBWEB is dependent on the order that instances are observed. On the third pass through the soybean data COBWEB segregated the soybean instances in a manner corresponding to their diagnostic condition. All orderings might not result in 'rediscovery' and the time until rediscovery may vary greatly depending on ordering. Chapter 6 discusses convergence time and related issues. It is important to note however that rediscovery is not important in the experiments that follow. A major impetus for COBWEB's characterization in terms of prediction is that it defines an objective dimension for evaluating the utility of clustering.

relationship, the induction test was repeated for each of the remaining 35 attributes. The results of these tests (including Diagnostic-condition) were averaged over all attributes and are presented in Figure 45. On average, correct prediction of attribute values for unseen objects levels off at 87% using COBWEB's classification tree.³⁸ Figure 45 also graphs the averaged results of the frequency-based method. Averaged results using this strategy level off at 74% correct prediction, placing it at 13% under the COBWEB classification strategy. However, these averaged results do not tell the whole story – the primary interest is in determining a relationship between attribute correlations and the ability to correctly infer an attribute's value using COBWEB classification trees.

To characterize the relationship between attribute dependence and inference ability it is necessary to introduce a measure of attribute dependence. The dependence of an attribute A_M on other attributes A_i is given as

$$\frac{\sum_i \sum_{j_i} P(A_i = V_{ij_i}) \sum_{j_M} [P(A_M = V_{Mj_M} | A_i = V_{ij_i})^2 - P(A_M = V_{Mj_M})^2]}{|\{i | A_i \neq A_M\}|} \quad (5-1)$$

This function is derived in much the same way as category utility, but it measures the average increase in the ability to guess a value of A_M given the value of a second attribute. If A_M is independent of all other attributes, A_i , then 5-1 equals 0 since $P(A_M = V_{Mj_M} | A_i = V_{ij_i}) = P(A_M = V_{Mj_M})$ for all A_i , and thus $P(A_M = V_{Mj_M} | A_i = V_{ij_i})^2 - P(A_M = V_{Mj_M})^2 = 0$.

Figure 46 shows the increase in correct prediction afforded by COBWEB's classification tree after 25 instances over the frequency-based method as a function of attribute dependence. Each point on the scatter graph represents one of the 36 attributes used to describe soybean cases. For example, after 25 instances diagnostic condition was correctly predicted 100% of the time using COBWEB's

³⁸ To emphasize, this is an *induction* task. As would be expected, when classifying previously incorporated objects, correct prediction of missing attribute values is generally 100%. See section 5.7 for further discussion.

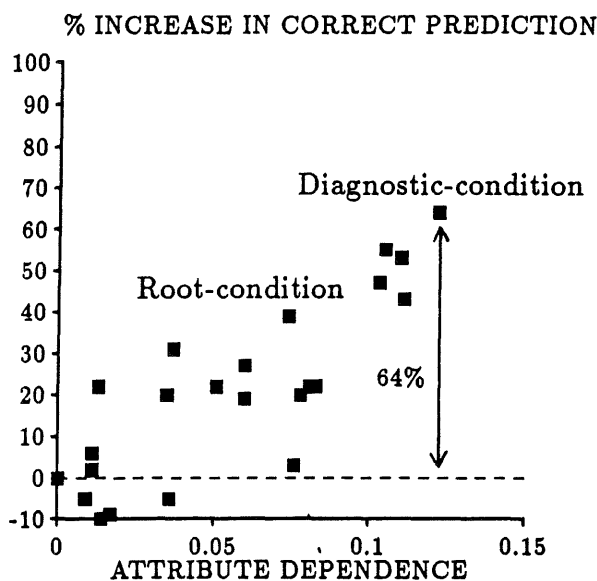


Figure 46

Increase in correct inference as a function of attribute dependence

classification tree, while 36% correct prediction was achieved using the frequency-based approach. Figure 46 shows the difference of 64%.

More generally, there is a significant positive correlation between an attribute's dependence on other attributes and the degree that COBWEB trees improve inference.³⁹ For example, Diagnostic-condition is most dependent on other attributes and prediction accuracy benefits most for this attribute. Prediction of attributes that approximate independence from other attributes does not benefit from classification and in the case of four attributes is less effective than the frequency-based approach.

A more general lesson of this analysis is that learning performance is as much (if not more) a function of the domain as it is of the learning algorithm. In some cases (e.g., Diagnostic-condition) learning is swift and correctness is high. In other cases, even after seeing half of the total objects, performance may not be as good as that obtained using a 'stupid' frequency-based approach. Domains must be

³⁹ The Pearson product-moment coefficient is 0.88, indicating a highly significant correlation.

Attributes	Domains	Example
age	(<18, 18-29, 30-42, 42-55, 55-68, >68)	?
sex	(M,F)	M
on thyroxine	(T,F)	F
query on thyroxine	(T,F)	F
thyroid surgery	(T,F)	F
query hypothyroid	(T,F)	F
query hyperthyroid	(T,F)	F
pregnant	(T,F)	F
sick	(T,F)	F
tumor	(T,F)	F
lithium	(T,F)	F
goitre	(T,F)	F
TSH	(normal,high)	normal
T3	(low,normal,high)	normal
TT4	(low,normal,high)	normal
T4U	(low,normal,high)	normal
FTI	(low,normal,high)	normal
TBG	(low,normal,high)	normal
Diagnostic Condition	(negative,sick-euthyroid,hypothyroid)	negative

Table 17

Sample thyroid disease cases

characterized before the advantages of a learning system can be properly assessed. These general findings are probably extendable to many inductive learning systems, although to date COBWEB's analysis is novel in this respect.

5.4.2 Learning to Diagnose Thyroid Disorders

The inference experiments were repeated for a second domain of 150 thyroid patient case histories.⁴⁰ Each patient was described by 19 attributes. Three diagnostic conditions were exhibited (with equal probability) in the data (hypothyroid, sick euthyroid, neither of these). The tree formed over the data by COBWEB is partially shown in Figure 47.

⁴⁰ This data was kindly supplied by J.R. Quinlan and taken from patients of the Garvan Institute of Medical Research.

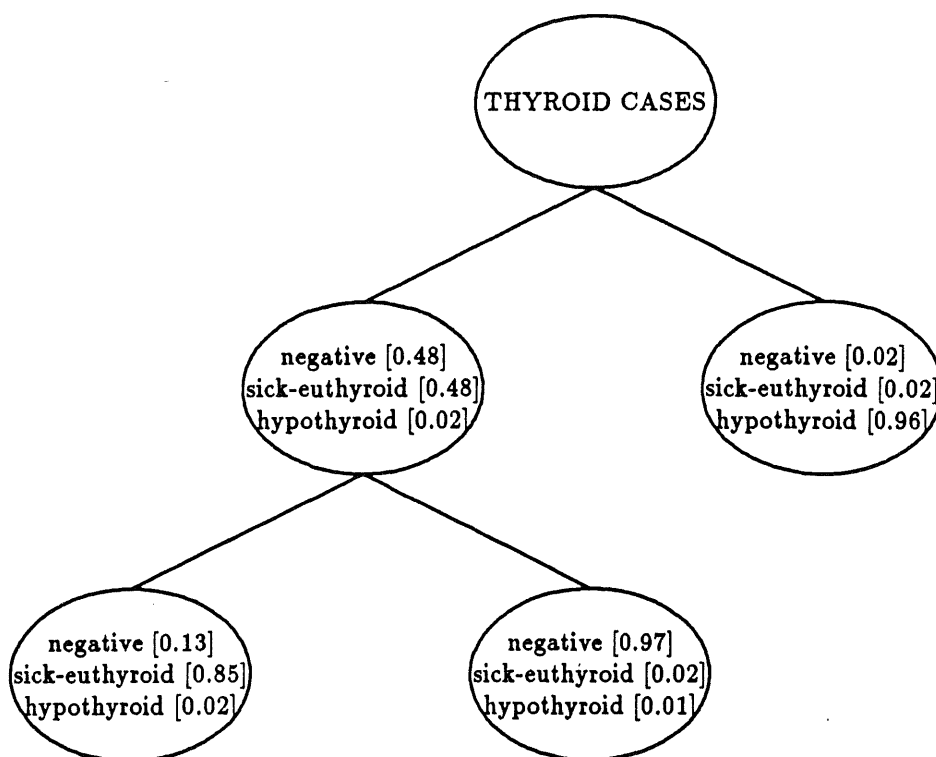


Figure 47

The top levels of a classification tree over thyroid disease cases

An experiment similar to that conducted for the soybean cases was conducted using the thyroid data. The graph of Figure 48 indicates that classifications rapidly became effective tools for diagnosis in this domain, with the percentage of correct predictions of Diagnostic-condition leveling off at 88%.

While prediction of Diagnostic-condition improves by 55% using COBWEB's tree, Figure 49 indicates that prediction averaged over all attributes increases only by 5% over the frequency-based approach.

Figure 50 shows the reason for this feeble average improvement. In general, attributes in this domain interact (statistically) less than those of the soybean domain. As Figure 50 shows, the relatively weak showing for COBWEB stems from the many attributes that are better predicted using the frequency-based approach. These attributes tend towards independence from other attributes.

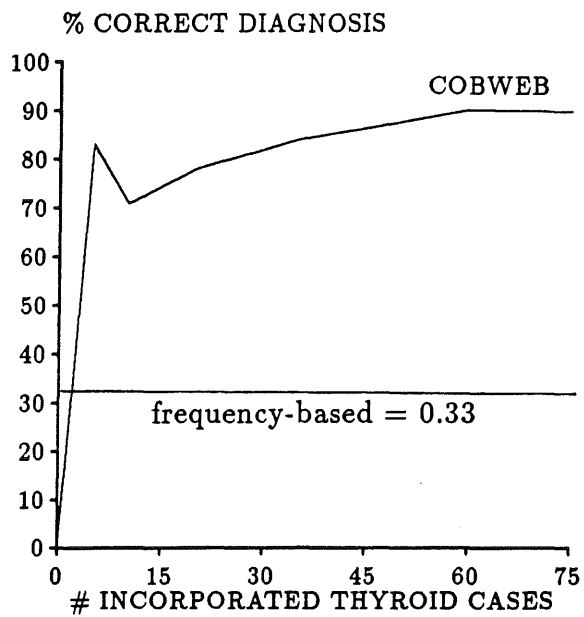


Figure 48

Diagnostic success with thyroid cases

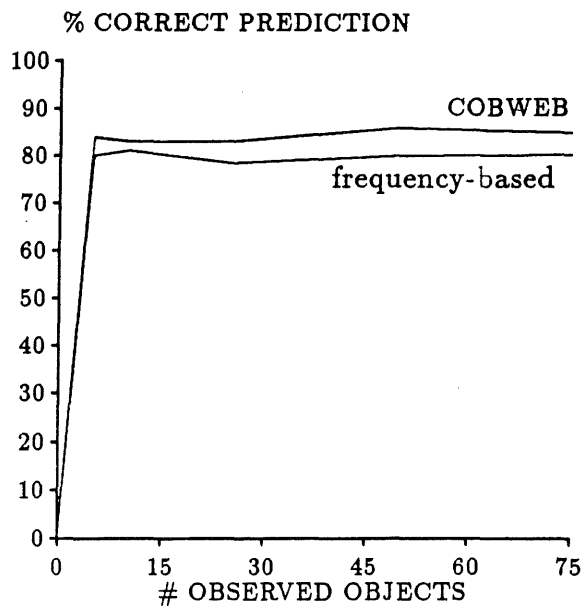


Figure 49

Success at inference averaged over all thyroid attributes

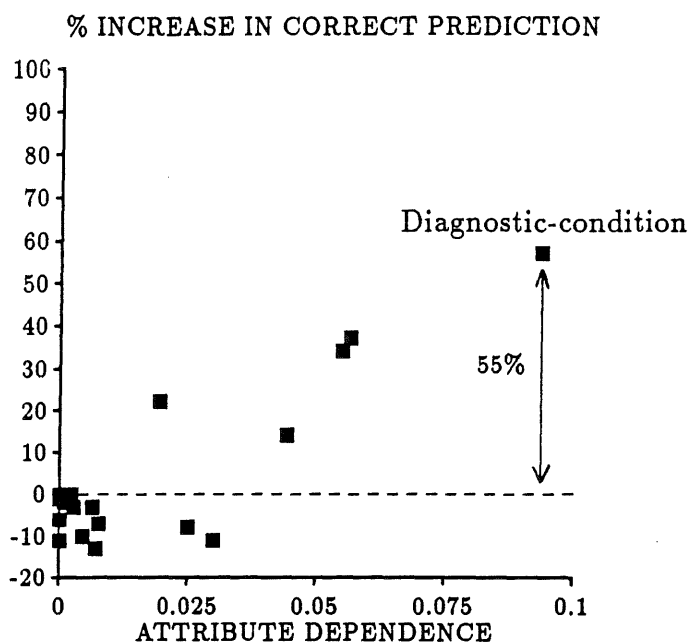


Figure 50

Increase in correct inference as a function of attribute dependence.

Experiments indicate that COBWEB is a viable means of organizing observations to support inference but at least initially, its utility must be qualified. Classification improves prediction accuracy with respect to attributes that participate in many data dependencies (e.g., Diagnostic-condition). In general, the efficacy of COBWEB's domain-independent approach stems from the observation that real-world domains tend to exhibit significant degrees of data dependence [MERV81]. However, there is room for improving prediction, particularly with respect to relatively independent attributes.

5.5 Inference Using Norms

In the experiments above, classification proceeded to a leaf before a prediction was made. Chapter 3 showed that the classification tree is a concise way of representing certain relationships and sub-relationships between attributes of a domain. For predicting the value of an attribute dependent on many other attributes, a

'deep' classification (e.g., to a leaf) is necessary if all dependencies are to be taken into account when making a prediction. However, if an attribute is independent of other attributes, traversal to a leaf may introduce spurious relationships that do not facilitate prediction. In fact, prediction of relatively independent attributes in the soybean and thyroid domains indicate that the frequency-based approach often Outperforms the use of a COBWEB classification tree.

The frequency-based method of prediction can be viewed as classification only with respect to the root of a tree. The root contains probabilistic information over all currently classified objects. Being limited to classification at the root, the best prediction for an attribute is its most frequently occurring value. Recalling discussion of norms, the collocations of values for an independent attribute will be maximized at the root. Therefore, the most frequently occurring value of an independent attribute will be a norm of the root.

In general, effective use of normative information can improve the cost and correctness of inference by demarcating when classification should cease and prediction should occur. In particular, the soybean and thyroid inference experiments were modified so that prediction could benefit from limited use of class norms.

5.5.1 Exploiting Norms in the Soybean Domain

COBWEB was trained on the same 25 soybean cases as earlier experiments. After building a classification tree over these objects, unconditioned normative values were identified. That is, collocation maximizing nodes were found for each attribute value. This computation was conditioned only on root-level statistics, which allowed limited nesting of normative values. For example, 'Root-condition = normal' is a norm for the top-most (root) class (true of 72% of the first 25 objects), while 'Root-condition = rotted' is a norm of a lower level node. Thus, some subnorms were generated. However, subnorms in the sense of Figure 41 were

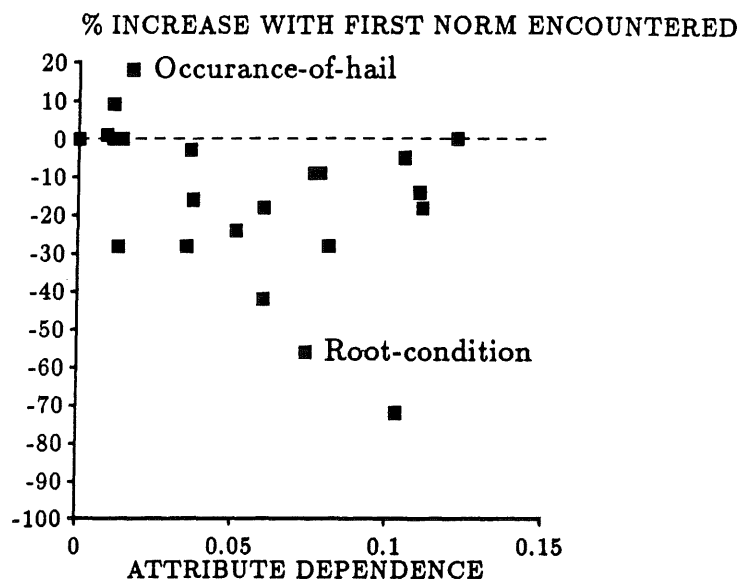


Figure 51

Difference between using and not using first soybean norm

not used for this experiment. Only unconditioned normative values were identified, not subnorms conditioned on lower (than the root) node statistics.

Using unconditioned norms, an initial experiment used the first normative value encountered during classification as the prediction of a missing attribute's value. Figure 51 shows the difference between correct prediction afforded by norms and prediction generated at leaves as a function of attribute dependence. For example, prediction of Occurrence-of-hail benefits from norms. Using norms, a 77% correct prediction is achieved over a 59% correctness rate without norms (i.e., classification to a leaf). Figure 51 shows the difference, which is 18%. On the other hand, prediction of Root-condition is 56% worse when the first encountered norm is used. The norm 'Root-condition = normal' is encountered immediately at the root. In general, prediction either benefits or remains the same when using the first encountered norm to predict relatively independent attributes. As attribute dependence increases, this method can be significantly worse than classification to a leaf (e.g., 56% worse).

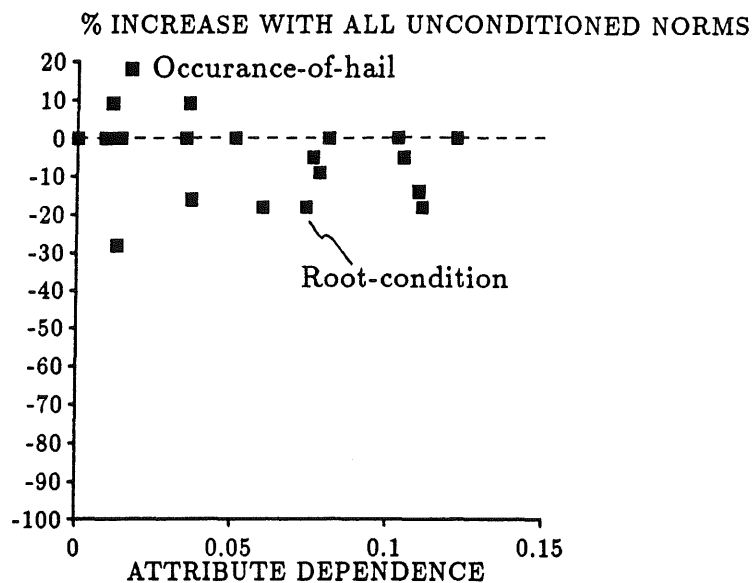
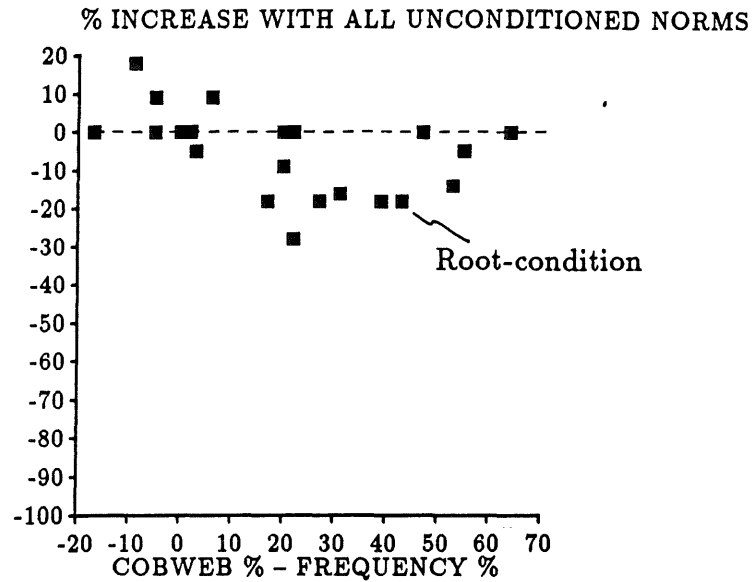


Figure 52

Difference between using and not using unconditioned soybean norms

The apparent need for deeper classification as a precursor to predicting more dependent attributes motivated a second experiment. Unlike the first experiment, limited use of subnorms was allowed (i.e., *all* norms were still unconditioned). A prediction was generated at the *last* unconditioned norm encountered before reaching a leaf.⁴¹ The results of this experiment are shown in Figure 52. In general, performance for most relatively dependent attributes is still not as good as achieved by classifying to a leaf. However, prediction of these attributes is still significantly better than is achieved by simply using the first encountered norm. For example, prediction of Root-condition is now 18% worse than achieved by classification to a leaf, as opposed to 56% worse when using the first norm. Prediction of most independent attributes remains unchanged, indicating the first and last encountered norms are the same. For many of these attributes collocation of all values is maximized at the root.

⁴¹ In general, identifying the last norm before a leaf necessitates classifying to a leaf and recalling the last norm encountered.

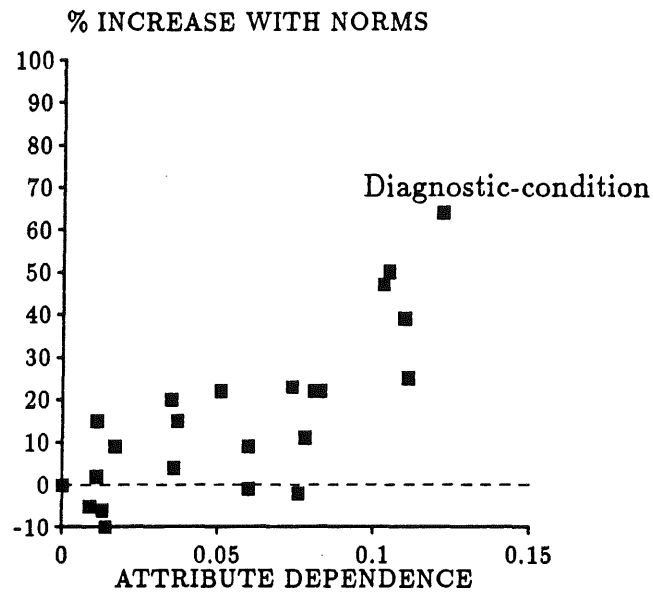


Difference between using and not using unconditioned soybean norms

In addition to graphing differences as a function of attribute dependence, results of the last experiment can be viewed as a function of how well classification (to leaves) stacked up against the frequency-based approach. This alternative view of the advantage of using norms is shown in Figure 53. This figure shows that in the soybean domain using unconditioned norms and the frequency-based approach tend to be superior (and inferior) to classification to leaves in the same situations.

While using norms and the frequency-based approach do relatively good and bad under the same conditions, the use of norms still beats the frequency-based approach overall. Figure 54 makes the difference between these guessing strategies explicit. The shape of this graph is reminiscent of Figure 45, which showed the difference between results obtained through classification to a leaf and the frequency-based approach.

Averaged results for these experiments are shown in Table 18 and indicate the cost effectiveness of using norms. Correct prediction based on norms averaged 84%, as compared to 87% (recall Figure 44) for classification to a leaf. While average



Increase in correct inference with norms over frequency approach

	Classification to a leaf	Unconditioned norms	First norm only	Frequency based
Correctness	0.87	0.84	0.78	0.74
Depth	3.09	0.35	0.12	0.00

Table 18

Averaged inference results in the soybean domain

correctness was nearly as good, the effort required in using norms is significantly less. On average, it required classification proceed to approximately 1/10 the depth (0.35 levels) that was required of classification to a leaf (3.09 levels).

5.5.2 Exploiting Norms in the Thyroid Domain

The cost effectiveness of using norms in the soybean domain is accentuated in the thyroid domain. The soybean domain illustrates that relatively independent attributes benefit most from normative values and these attributes dominate the thyroid domain. Figure 55 shows the results of using the first encountered norm for

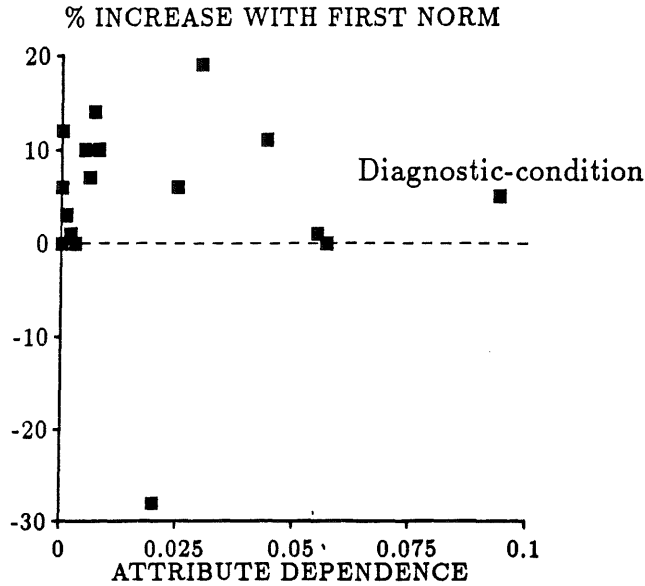


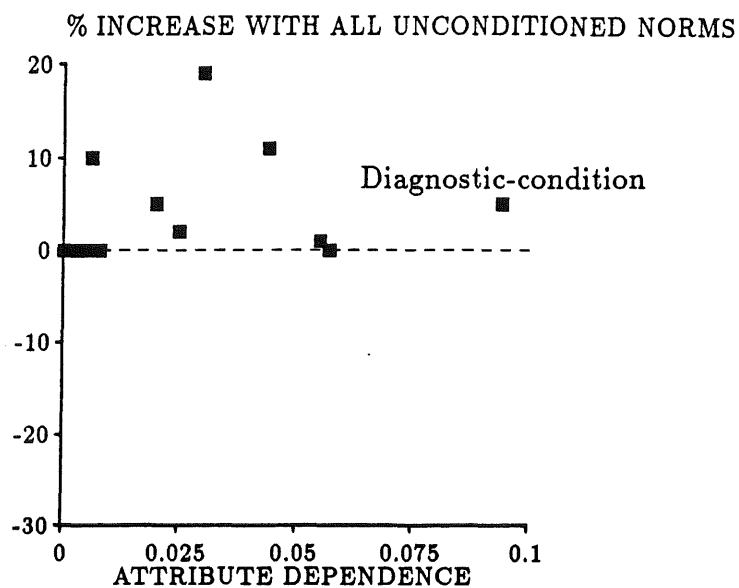
Figure 55

Difference between using and not using first thyroid norm

prediction. Like the soybean domain, prediction of independent attributes tends to benefit most. However, unlike the soybean domain, dependent attributes benefit as well, though not to the same extent. There is an average increase in prediction correctness using the first encountered norm over using no norms at all.

Results from using all unconditioned norms (i.e., last encountered) are shown in Figure 56. In this case there is a slight decline in average performance as compared to using the first norm. This is due to a drop in correct predictions of independent attributes. However, note that overall accuracy remains higher than using no norms whatsoever.⁴² As Table 19 indicates, in the thyroid domain prediction benefits from norms not only in terms of cost (i.e., depth of classification), but in correctness as well.

⁴² The shape of this scatter graph is roughly maintained after changing the horizontal dimension to be the difference between performance obtained from classification versus the frequency-based approach.



Difference between using and not using unconditioned thyroid norms

	Classification to a leaf	Unconditioned norms	First norm only	Frequency based
Correctness	0.85	0.87	0.89	0.80
Depth	4.81	0.49	0.35	0.00

Table 19

Averaged inference results in the thyroid domain

5.5.3 Summary

Using norms can improve the cost effectiveness of prediction. Demonstrations in the soybean and thyroid domains indicate that the simple rule of using the first norm encountered during classification improves prediction with respect to relatively independent attributes. In general, this improvement is tempered by a symmetric decrease in accuracy with respect to dependent attributes, although prediction of the (relatively few) dependent attributes of the thyroid domain actually

increased as well. Averaging over all attributes indicates that accuracy using norms approximates the accuracy achieved without norms, at significantly less cost.

The norms used in these experiments have been called 'unconditioned' norms, since their identification is trivially conditioned on root node statistics. They have provided a simple mechanism for demonstrating the promise of using normative information. However, it seems apparent that a method taking greater advantage of nested norms (conditioned, as well as unconditioned) can improve the cost effectiveness of prediction of more dependent attributes by stipulating when it is best to predict the value of an attribute. Any such method must trade cost and correctness. Such a method is not described here, but it is left as future work.

5.6 Conceptual Clustering and Learning from Examples

The soybean and thyroid data strongly suggest that COBWEB captures the important inter-correlations between attributes and that it summarizes these correlations at classification tree nodes. In doing so, COBWEB promotes inference of attributes roughly in proportion to the degree that they participate in correlations. This is in contrast to learning from examples, which seeks to maximize correct prediction with respect to a single 'teacher' selected attribute. Thus, the performance task associated with COBWEB (and implied for conceptual clustering, generally) generalizes the performance task of learning from examples. However, this generality may come at the expense of correctness with respect to individual attributes. This observation suggests a way of obtaining a rough theoretical upper bound on COBWEB's inference ability.

5.6.1 ID3 and COBWEB

In order to demarcate an upper bound on COBWEB's performance, prediction accuracy stemming from a single COBWEB classification tree was compared

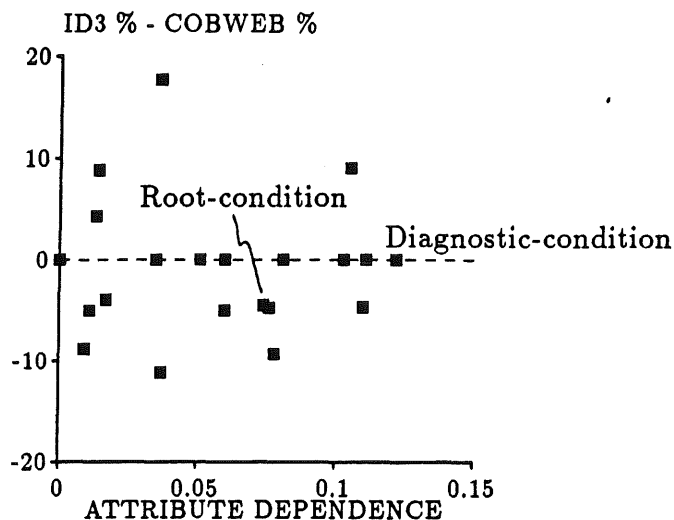


Figure 57

A comparison of ID3 (no chi-square) and COBWEB

to results obtained from a reconstruction of Quinlan's (1983) learning from examples program, ID3. The program, ID3', builds *decision trees* to distinguish object classes.⁴³ Specifically, for *each* of the 36 attributes of the soybean domain, the training set (of 25 instances) observed by COBWEB was used to train ID3'. In each case, the values of the attribute were treated as 'teacher' imposed classes. Thus, ID3' built one decision tree to distinguish the various diagnostic conditions, a separate tree to distinguish seed conditions, and a distinct tree for each subsequent attribute. These decision trees were used to predict the appropriate attribute values in the remaining unclassified soybean cases.⁴⁴

Two variations on this basic experiment were conducted. In the first, classification to a leaf of COBWEB's tree preceded prediction. Results obtained by this method were compared to a version of ID3' that did not use the chi-square measure

⁴³ ID3' does not include 'windowing' and other efficiency enhancements.

⁴⁴ In the context of ID3, the single domain of soybean case histories can be viewed as 36 individual 'domains'. ID3 is being run for each attribute, not simply Diagnostic-condition. The measure of data dependence (5-1) can be interpreted as a measure of domain complexity. The 36 'domains' implicit in the soybean case histories are well distributed across the complexity space defined by 5-1.

	ID3' (no chi-square)	COBWEB (no norms)	ID3' (chi-square)	COBWEB (unconditioned norms)
Correctness	0.87	0.87	0.88	0.84

Table 20

Averages from the ID3 experiments

to determine when to stop expanding the tree. Decomposition ceased when all objects classified under a node had the same value along the appropriate attribute, but there was no check using chi-square of whether the best divisive attribute was a statistically useful classifier. Figure 57 gives the *differences* between correct prediction using each ID3' decision tree and the single COBWEB classification tree. For example, COBWEB's tree predicts Root-condition correctly 100% of the time, while the ID3' decision tree for this attribute yields 96% correctness, giving a difference of -4%.

On average, correctness afforded by the COBWEB classification tree is comparable to that afforded by the 36 ID3' decision trees. However, this statement must be qualified. This variation of ID3' does not include protections against 'exceptional' objects (i.e., the chi-square measure). A second experimental variation used a version of ID3' that applied chi-square to control tree construction. The use of chi-square in ID3' and norms in COBWEB are motivated by the same principle of avoiding over-specialization.

5.6.2 The Upper Bound Supplied by Learning from Examples

Table 20 shows averaged results of the ID3' and COBWEB experiments. Overall, COBWEB's single tree approximates the predictive ability of ID3's 36 trees. In fact, Figure 57 shows that COBWEB trees facilitate better prediction than a corresponding ID3' decision tree for many (9) attributes. However,

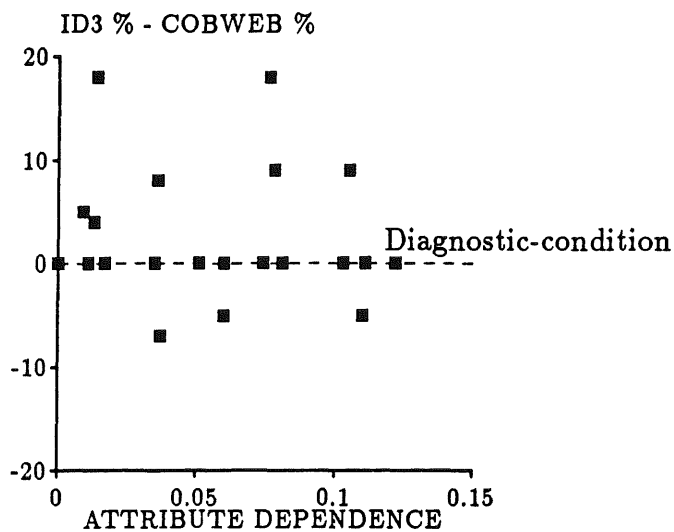


Figure 58

A comparison of best scores obtained by ID3' and COBWEB.

Figure 58 gives a slightly different picture of performance that takes into account ID3' using chi-square and COBWEB using unconditioned norms. Like Figure 57, Figure 58 shows the difference between ID3' correct prediction and COBWEB's. However, in this experiment the best score obtained by either version of ID3' (i.e., using and not using chi-square) is compared against the best score from either version of COBWEB (i.e., with and without norms). Note that ID3' does as well or better along all but three attributes. Hopefully, this analysis approximates a fair comparison between the theoretical best scores obtained using ID3' and COBWEB trees.

Figure 58 indicates that ID3' provides a rough upper bound on correct prediction with respect to most attributes, as well as on the average (1% difference). However, three caveats preclude the possibility of any strong conclusions at this point. First, comparisons of COBWEB's classification tree and ID3 decision trees may be unfair; ID3 trees discriminate based on a *single* attribute at decision points (i.e., they are *monothetic* classifiers), while the use of category utility as a matching function makes COBWEB trees essentially *polythetic* and therefore more sensitive

to attribute inter-correlations. The advantage of polythetic classification may be magnified by the small data set.

To summarize, there may be learning from examples systems that further improve upon COBWEB's behavior. However, the possibility of improved learning from examples behavior may be counteracted by a more sophisticated method of using nested norms in COBWEB. To some extent, the choice between selecting classification to a leaf or classification to the last unconditioned norm is one way of approximating the performance of a more robust method. Finally, irrespective of the polythetic/monothetic issue, chapter 3 pointed out that a weakness of strict trees is that if 'orthogonal' inter-dependencies exist in the data, only some may be isolated at tree nodes. Not all dependent attributes can be equally rewarded. This may be a fundamental limitation on the performance of both ID3' and COBWEB with respect to *induction* tasks.

Last, the use of normative values in COBWEB and chi-square in ID3 are motivated by similar concerns – it may not be desirable to classify an object too precisely. Recent results by Quinlan [QUI87A] and Michalski [MICH87] indicate that it is possible to simplify decision trees and concept descriptions and only decrease slightly or actually improve prediction accuracy over unseen cases. However, COBWEB's results using norms imply a dimension along which the benefits of simplification vary. While extending COBWEB's results to the issue of concept simplification is left as future work, the advantages of simplification probably vary as a function of certain domain characteristics (e.g., attribute dependence).

5.7 Induction and Deduction

Prediction emerges naturally from classification. The experiments of this chapter illustrate this point, but they focus exclusively on classifying objects that were not used in classification tree construction. As with any inductive learning

program, COBWEB assumes that the patterns (rules, correlations) expressed over the observed objects approximate the patterns of the whole environment. For example, having seen the objects

- (1) $\{A_0 = 0, A_1 = 0, A_2 = 1, A_3 = 1\}$,
- (2) $\{A_0 = 0, A_1 = 0, A_2 = 0, A_3 = 0\}$,
- (3) $\{A_0 = 1, A_1 = 1, A_2 = 1, A_3 = 0\}$, and
- (4) $\{A_0 = 1, A_1 = 1, A_2 = 0, A_3 = 1\}$,

COBWEB may form object classes, $N_1 = \{1, 2\}$ and $N_2 = \{3, 4\}$, corresponding to cases where $A_0 = 0, A_1 = 0$ and $A_0 = 1, A_1 = 1$, respectively. Suppose that this grouping is used to predict the value of A_1 for a new object, $\{A_0 = 0, A_1 = ?, A_2 = 0, A_3 = 1\}$. Given COBWEB's prior grouping into N_1 and N_2 and the classification procedure used throughout this chapter, the values of A_2 and A_3 offer no help in classifying the new object. However, the new object is classified as a member of N_1 by virtue of $A_0 = 0$. At this point the value of A_1 can be predicted to be 0 since this is the case with all members of N_1 . Of course this prediction may be wrong, but underlying this prediction is the assumption that because $A_0 = 0 \rightarrow A_1 = 0$ (or more precisely, $A_0 = 0 \rightarrow N_1 \rightarrow A_1 = 0$) is true of the observed objects, it will be true of all objects of the domain. This is an example of *inductive* reasoning.

More generally, one type of inductive inference takes the form:

- if** a property, P , is true of each member of an object set, $O = \{O_1, O_2, \dots\}$
 (i.e., $P(O_1), P(O_2), \dots$)
- then** P is true of all objects *if the set of all objects is a proper superset of O .*

If O is not the set of all objects then asserting P is true of an arbitrary object (member or nonmember of O) may prove false. However, as O closer approximates the set of all objects, the higher the expectation that such an assertion will be true (i.e., the less the 'inductive leap'). Finally, if O equals the set of all objects then an

assertion of $P(O_i)$ must be true for any O_i . The fact that $P(O_i)$ can be asserted with certainty is an example of *deductive* inference.⁴⁵

Consider the relationship between induction and deduction using ID3 (without chi-square) decision trees as an example. ID3 builds a decision tree that distinguishes two or more classes, lets say C and $\neg C$. A decision rule, R , that distinguishes C from $\neg C$ can be generated by taking the disjunction of all paths leading to a leaf labeled by C . C and R can each be regarded as predicates that are true of a particular instance or not. Over the observed objects, $R(x) \equiv C(x)$, but in using the decision tree we are typically only interested in $R(x) \rightarrow C(x)$. In the terminology developed above $P(O_i) \equiv [R(O_i) \rightarrow C(O_i)]$. If the observed objects are a subset of all objects, this leads to inductive reasoning, i.e., $R(O_i) \rightarrow C(O_i)$ may or may not be true for all objects and $C(O_i)$ may be incorrectly asserted of an object O_i if $R(O_i)$. However, if the decision tree classifies all possible objects, the truth of $C(O_i)$ is ascertained deductively – with certainty.

Like ID3, the ‘inductive leap’ using COBWEB classification trees decreases as the number of observed objects increases. As a result, prediction using COBWEB’s classification trees becomes increasingly accurate. After all objects of a domain are observed, prediction is ideally a deductive process. As evidence, consider that prediction was 100% correct for every attribute over the 25 soybean case histories *used to build the tree* in the experiments reported in this chapter.

Despite the perfect performance over observed soybean case histories, some qualifications apply to the claim that prediction (via classification) eventually becomes deductive using COBWEB. In particular, COBWEB may require seeing

⁴⁵ There are two generally accepted deductive rules [MEND79]. Informally stated, the first is ‘if a property is true of all objects then it is true of each one of them.’ The second, *modus ponens*, states that if $p \rightarrow q$ and p are true then q is true. Philosophy traditionally seems to regard induction as any process that employs the inverse of either deductive rule [SKYR75], while others [CHAR85] regard induction as using the inverse of the first deductive rule and *abduction* as inference using the inverse of *modus ponens*. However, comparisons are complicated by the connection drawn by some between abduction and the less formal notion of *causality*.

in the case of relatively independent attributes, and it hinted at an approach for cost effective inference with respect to all attributes. COBWEB's performance was also compared that of ID', a system that learns from examples. COBWEB attempts to maximize the correctness of prediction with respect to all attributes, while learning from examples attempts to maximize correctness with respect to a single attribute. Comparisons between COBWEB and ID3' indicate that a single COBWEB classification tree approximates the abilities of multiple ID3' decision trees for many attributes. However, ID3' still provides an upper bound. The general lesson is that when there is a known attribute for which prediction should be maximized, conceptual clustering should not serve as a replacement for learning from examples.

The methodological biases demonstrated in this chapter are severalfold. Most important is the observation that the performance of inductive learning programs must be qualified. The statement that a system achieved accuracy 'x' in time 't' in a domain conveys *no* information *per se*. Results must be compared to alternative methods to put results into perspective. Furthermore, an attribute dependence score was used to characterize the difficulty of COBWEB's prediction tasks (or 'domains' for ID3). As demonstrated, prediction accuracy is quickly achieved for some attributes, while perfect prediction may never be obtained for others. While there may be problems with attribute dependence as a general measure of domain difficulty, it is important to note that COBWEB's analysis follows Simon's [SIMO69] suggestion that domains be characterized along with AI systems. This methodological bias is novel with respect to current practices in AI.

While many domains exhibit significant regularities in data, the ability to uncover these regularities may be hindered by a number of factors. For instance, one difference between COBWEB and other conceptual clustering systems is that it is incremental. The inability to examine all instances simultaneously can be a

significant barrier to learning. The next chapter analyzes the impact of incremental processing.

Chapter Acknowledgements

Prediction of unknown attributes was selected as a performance task with the help of Dennis Kibler. Discussions with Rick Granger and Dave Ruby led to the link between the performance tasks of conceptual clustering and learning from examples. Jeff Schlimmer and Nick Littlestone suggested the comparison between COBWEB and ID3'. The experimental methodology demonstrated in this chapter is due to the influence of Jeff Schlimmer, Ross Quinlan, Rick Granger, Dennis Kibler, and Steve Hampson. Graphs shown in this chapter were produced using a graph-generation program written by Jeff Schlimmer.

CHAPTER 6

COBWEB as an Incremental Learner

6.1 Chapter Overview

COBWEB is an incremental conceptual clustering system. As such, it has been designed to accept objects one at a time; objects of the environment are not assumed to be present all at once. This is a necessary property of systems that are to be usefully applied in many real-world domains. Moreover, while COBWEB stores all previously observed instances, only a small fraction of them are reexamined when a new observation is incorporated. This characteristic keeps incorporation costs down and insures that the system can rapidly update memory to reflect new stimuli.

A bias of this dissertation is that rapid memory update is a major constraint on the design of many incremental systems. In COBWEB, this constraint motivated a hill-climbing search strategy through the space of possible partitions and classification trees. While many systems can be made to *behave* incrementally – incorporate objects observed – search intensive strategies may not be practical in these situations, since they require updating a frontier of hypotheses or examining a list of previously observed instances.

Along with the cost advantages of a hill-climbing implementation of incremental processing comes some disadvantages. By their nature, search-intensive systems are more likely to find the ‘best’ hypotheses according to some criteria (under the assumption that sufficient time is available for search). That is, a frontier of hypotheses can be maintained until one emerges as the optimal. A hill climber maintains only one hypothesis; early in the learning process a hypothesis may be

kept that later proves to lead to a nonoptimal solution. However, a suggestion of chapter 2 is that a hill climber can overcome some of these limitations by using operators that move bidirectionally in the search space. This allows the system to approximate the effects of backtracking through operator application. COBWEB implements such a strategy by using divisive (splitting) and agglomerative (merging) operators in tree construction. Last, bidirectional movement may require that more objects be observed by the system if it is to find the same solutions as a search-intensive system. The latter strategy generates (and keeps) all hypotheses consistent with a new object. This information can be recovered through backtracking (e.g., as in depth-first search) or it is maintained simultaneously (e.g., as in breadth-first search). 'Recovering' alternative hypotheses in a bidirectional hill climber requires that operators be 'fired' by additional observations.

Motivated by an interest in alternative designs for incremental systems, Schlimmer and Fisher [SCH86A] propose three criteria for evaluating incremental systems. These criteria (adapted for conceptual clustering) are:

- the *cost of incorporating* a single instance into a classification,
- the *quality of learned classifications*, and
- the *number of objects* required by a system to converge on a stable classification.

For incremental systems in general, incorporation cost should be low, thus allowing real-time update. However, this may come at a cost of learning lower quality classifications and/or requiring a larger sample of objects to find a good classification than a similarly intended nonincremental and search-intensive method. This chapter characterizes COBWEB in terms of these criteria and shows the system to be an economical and robust learner.

6.2 Cost of Assimilating a Single Object

A desirable property of incremental systems is the update be performed quickly. This section shows that COBWEB keeps update costs low. The cost of incorporating a new object into an existing classification tree can be computed by

$$\text{cost} = (\text{number of nodes an object is compared to}) \times (\text{comparison cost}). \quad (6-1)$$

To simplify the analysis, cost is first computed without regard to the promote operator.

In COBWEB, the cost of comparing an object to a single node involves incrementing appropriate counts of the node and evaluating the entire partition of which the node is a member. Evaluating an entire partition makes the comparison cost linear with respect to the average number of nodes in a set of siblings (i.e., the average branching factor). Let B be the average branching factor. If A is the number of defining attributes and D is the average number of values per attribute, then

$$\text{comparison cost} = O(BAD). \quad (6-2)$$

In addition to testing an object with respect to existing nodes, a singleton class with the object as sole member is evaluated (one comparison), the result of merging the two best hosts is evaluated (one comparison), and the result of splitting the best host is evaluated (B comparisons). The cost of these additional tests is *added* to that of testing existing nodes, so that 6-2 remains a legitimate upper bound approximation of comparison cost.

The number of nodes to which an object must be compared is approximated by the product of the average number of nodes at each level (B) and the average

depth of the classification tree. The depth of a tree that classifies n objects is approximately $\log_B n$. Thus,

$$\text{number of nodes} = B \log_B n. \quad (6-3)$$

Substituting 6-2 and 6-3 into 6-1, the approximate cost of assimilating a single object into a classification hierarchy is given as

$$\text{cost} = O(B^2 \log_B n \times AD), \quad (6-4)$$

where A = number of defining attributes,
 D = average number of values per attribute,
 B = average branching factor of the tree, and
 n = number of objects classified by the tree.

Thus, the cost of incorporating a new object into an existing hierarchy is logarithmic with respect to the number of previously seen instances and quadratic with respect to the average branching factor.

This analysis does not take into account the 'promote' operator. Recall that during object incorporation, some nodes may be tested for uselessness. The increased work required for promoting stems from the test for uselessness and not from the actual operation of promoting a node. The uselessness test requires ascending the classification tree from the node in question to the root. The ascent requires comparing the node's prototype with approximately B nodes at each level. The most levels that are ascended is the depth of the tree, $\log_B n$. At worst, each node tested for uselessness triggers a search of $B \log_B n$ comparisons. Theoretically, at any given branch of the classification tree with B nodes, $B - 1$ nodes may trigger the test for uselessness. Testing these $B - 1$ nodes of *one* tree level may trigger $(B - 1) \times B \log_B n = O(B^2 \log_B n)$ comparisons. This can happen for

each level of which there are about $\log_B n$, so the total cost that may be incurred is about $O(B^2 \log_B^2 n)$ comparisons, thus increasing the upper bound on incorporation cost. However, empirically $O(B^2 \log_B^2 n)$ does not seem to be a tight bound on the worst case performance, and it is certainly not a good reflection of average cost. Unreachable nodes are quickly filtered out and few appear to be created to begin with. Empirically, the number of nodes tested for uselessness at a level does not exceed two, but is generally zero. From this observation it follows that a tighter bound on the cost of promotion is $O(B \log_B^2 n)$. This still alters 6-4 as an upper bound on update cost in the worst case. However, under the assumption that zero objects are generally tested for promotion, 6-4 remains a good average-case approximation.

The cost of adding a single object to a classification tree is $O(B^2 \log_B n)$ for COBWEB, where n is the number of previously seen instances and B is the average branching factor of the classification tree. Fortunately, the branching factor does not appear to be dependent on the *number of objects* in a domain, but is dependent on *regularity inherent in the environment*. Further, the branching factor is not bounded by a constant (as in CLUSTER/2 [MIC83A]) or the average number of values per attribute (as in RUMMAGE [FIS85A] or DISCON [LANG84]). Tests in a variety of domains show that the branching factor ranges from two to five. In any case, the cost of adding a single object in COBWEB is significantly less expensive than rebuilding a classification tree for a new object using a nonincremental clustering method such as CLUSTER/2 which Fisher and Langley [FIS86A] have shown requires polynomial time of degree B .

6.3 The Quality of Classification Trees

Unlike incremental systems like UNIMEM [LEBO82] and CYRUS [KOL83A], COBWEB explicitly attempts to form classifications where the first level is an

optimal partition of an object set. Although node splitting and merging reduce the sensitivity of COBWEB to initial sample skew, and thus aid convergence on optimal partitions, *no* pure hill-climbing approach completely eliminates susceptibility to becoming trapped in local optima.

The objective of this section is to explore COBWEB's ability to find optimal partitions versus suboptimal partitions. Ideally, such an analysis should vary a number of dimensions to determine what effect this variance has on system performance. For example, the prediction experiments of chapter 5 took one approach to testing a system under different conditions; the measure of attribute dependence that was used can be viewed as a measure of task or domain 'difficulty'. This measure was applied in two natural domains and variance in this measure was highly correlated with attribute prediction correctness. The hope is that this correlation extends or generalizes to other domains. However, this strategy may be difficult to adapt to the present question about optimality. In particular, the optimal partition of an arbitrarily chosen domain may be very difficult to uncover by human or machine. If the optimal partition cannot be determined *a priori*, there is no way of telling whether COBWEB uncovered it or not either. Accordingly, the methodological approach of this section is to use artificially constructed domains. These domains are constructed in a manner that makes the optimal partition easily determined. While natural domains were appropriate for earlier studies, the pliability of artificial domains make them better suited for demonstrating the range of a system's behavior.

COBWEB was tested in four artificial domains. Figure 59 shows state machines representing these domains. Each state machine represents a domain whose objects it recognizes. Each object, regardless of domain, is represented by attributes A_0 through A_3 . For example, domain 4 contains ten instances, one of which is $\{A_0 = 0, A_1 = 0, A_2 = 4, A_3 = 4\}$. Domains 1, 2, and 3 contain four, six,

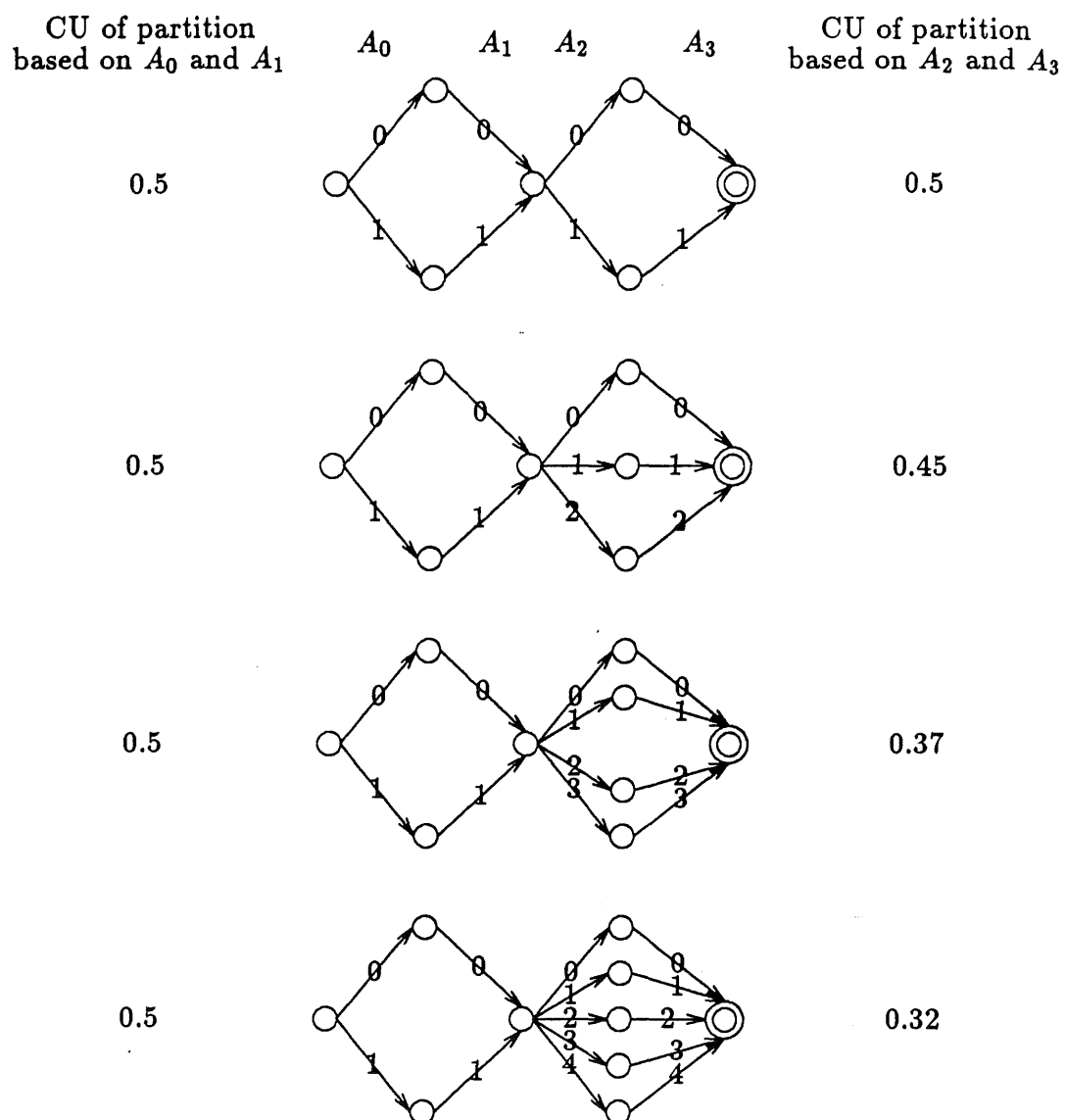


Figure 59

Domains with global and local optimums

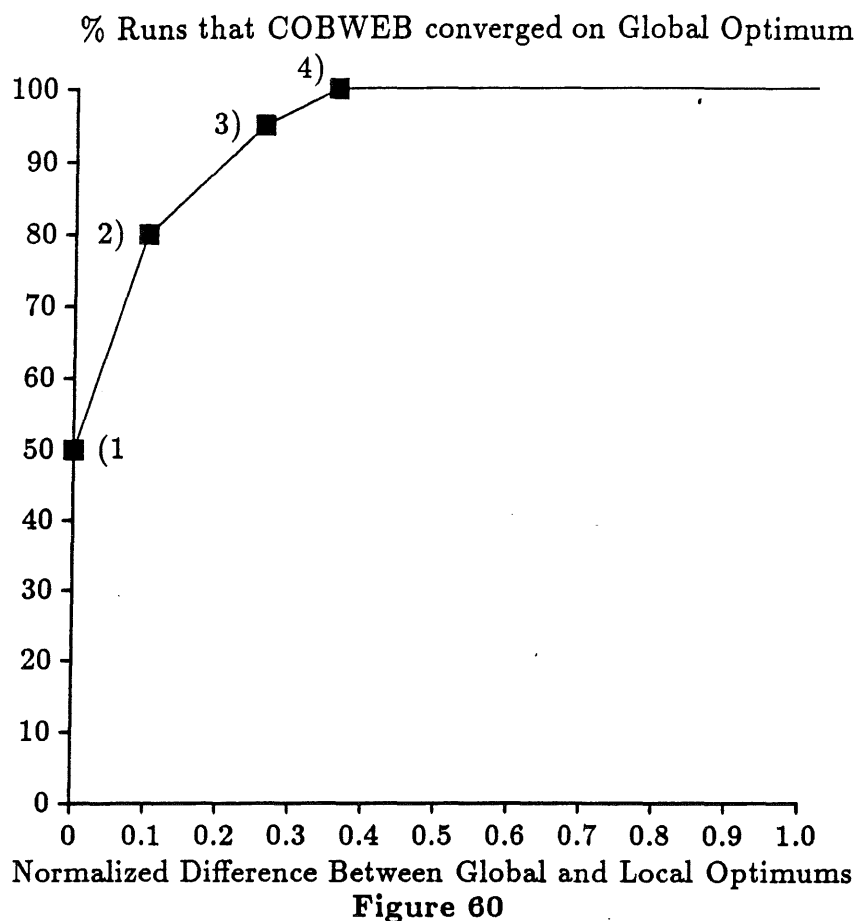
and eight objects, respectively. A state machine representation was chosen since it is compact and gives a pictorial view of the correlation between attributes. This view of attribute correlations also makes it easy to identify the optimal partition of each domain. Note that attributes A_0 and A_1 are codependent, as are A_2 and A_3 . This implies that a partition based on either pair of attributes will lead to

some increase in prediction ability as measured by category utility; category utility scores for both of these partitions are given in the figure for each domain. While there are codependent attribute pairs, note that A_0 and A_1 are independent of A_2 and A_3 , and vice versa. Thus, a partition based on any crossover between these attribute pairs will not capture a useful correlation and will not result in a higher category utility score than the two partitions based on the separate attribute pairs.

In each domain, the optimal partitioning of objects is a segregation based on the values of attributes, A_0 and A_1 , including domain 1 in which there is a tie. Partitioning based on attributes A_2 and A_3 forms a partition of lesser quality – a local optimum. Each domain was constructed so that there was a global and local optimal partitioning. The difference between the quality of the global and local optimum (in terms of category utility) was systematically varied from domain 1 with the least difference to domain 4 with the greatest difference.

COBWEB was run 20 times on random samples of 50 objects from each domain. Since none of the four domains has more than 10 distinct instances, each test required multiple observations of the same objects. The graph of Figure 60 shows the results of these runs. The vertical scale gives the percentage of runs in which the optimal partition was discovered. The horizontal scale gives the difference between the category utility of the optimal and local optimums (normalized to lie in $[0, 1]$).⁴⁶ The graph indicates that as the distance between global and local partitions grows, the possibility of becoming trapped a local optimum rapidly diminishes. COBWEB's inability to converge on optimal partitions in extreme cases is a direct result of its hill-climbing strategy. A search-intensive method would typically discover the optimal partition in all situations. However, since category utility measures the degree that a partition promotes correct prediction of object

⁴⁶ Distance was normalized by taking the optimal score, subtracting by the local score, and dividing by the optimal. A normalized score of 0 indicates the 'global' and 'local' optimum are tied, while a score of 1 indicates there is only one optimum or peak in the domain.



Convergence on optimal partitions

properties (i.e., the expected number of correctly predictable properties), the graph shows that COBWEB finds the global optimum when it is most important to do so (i.e., when there is most at stake in terms of correct inference). COBWEB will stumble into local optimum only when there is little lost in terms of inference ability.

6.4 Number of Objects Required For Convergence

COBWEB converges on classification trees in which the first level tends to be a global optimum. In this section, the system is discussed in terms of a third

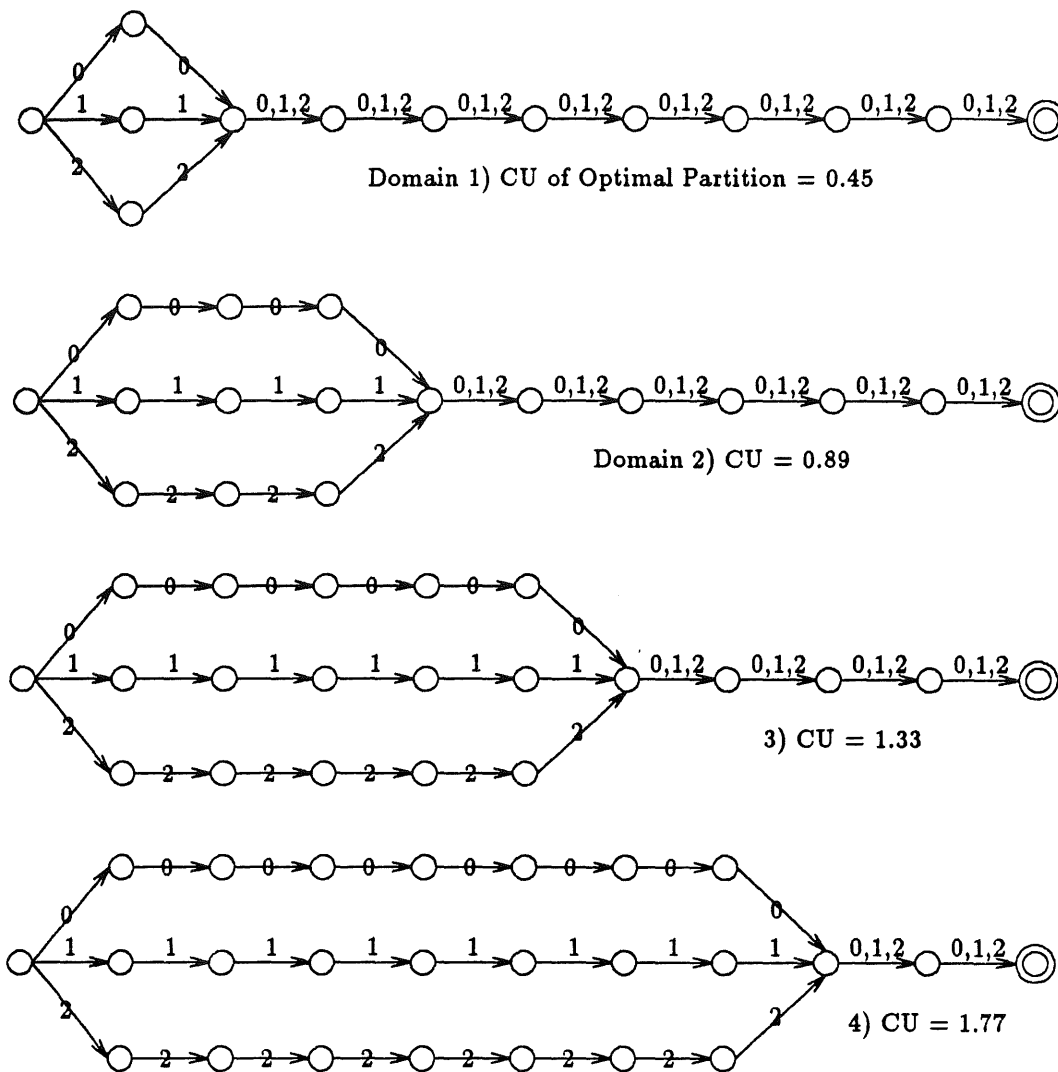


Figure 61

Domains used to test convergence time

criterion for evaluating incremental methods: the number of objects required to converge on a 'stable' (global or local optimum) partition.

Again, four artificial domains were used to test COBWEB. These domains are represented by the state machines of Figure 61. The number of objects in domains 1 through 4 are 19683, 2187, 243, and 27, respectively. Members of all domains are represented along attributes A_0 through A_9 . An example of an object from domain 2 is $\{A_0 = 1, A_1 = 1, A_2 = 1, A_3 = 1, A_4 = 0, A_5 = 2, A_6 = 2, A_7 =$

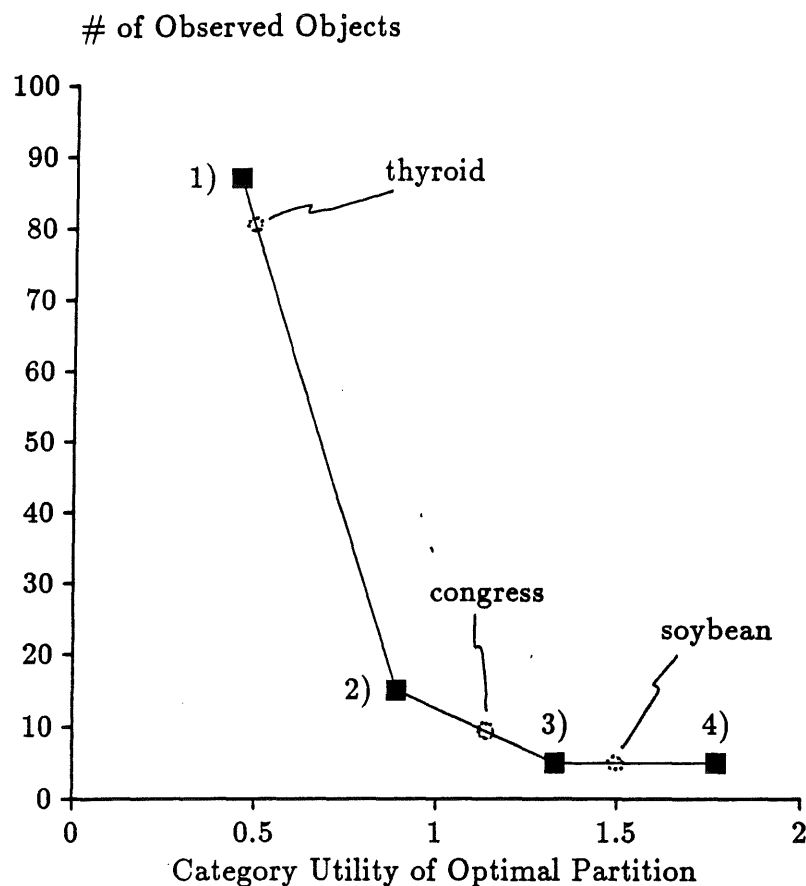


Figure 62

Number of objects required to converge

1, $A_8 = 0, A_9 = 0$ }. Note that in this example all values of the initial attributes, A_0 through A_3 , are the same. More generally, the four domains were selected so the optimal partition of each domain is unambiguous and easily visualized. In each case the optimal partition is one based on the values of the initially correlated attributes (e.g., A_0 and A_1 for domain 1, A_0 through A_3 for domain 2, etc.) – all remaining attributes are mutually independent.

The domains of Figure 61 systematically differ in terms of the quality of the optimal partition. The question that this section investigates is whether the ease with which the optimal partition can be incrementally discerned varies with the category utility (i.e., absolute quality) of the optimal partition.

COBWEB was run on 5 random orderings from each of the four domains. During learning, 100 objects were intermittently and randomly selected from the domain being learned and classified (but not incorporated) with respect to the classification thus far formed by COBWEB. If the top-most partition of COBWEB's classification tree segregated the sample in the same manner as the optimal partition of the environment as a whole, the two partitions were regarded as equivalent.⁴⁷

Figure 62 shows the results of this experiment. As the quality of the optimal partition grows, the fewer objects COBWEB required to converge on the optimal partition, or more accurately, a partition equivalent to the optimum. Inversely, as the quality decreases the number of objects required for convergence appears to increase exponentially. While a search-intensive method would probably exhibit a similar curve, the rate of increase would be considerably less as the quality of the optimal partition decreased.

While COBWEB may require many objects to stabilize on a partition, it appears to converge rapidly in domains of significant regularity.⁴⁸ To put some of the previously examined domains in context, the partitions (first level of the classification trees) formed for the soybean, congressional, and thyroid domains measured 1.50, 1.20, and 0.50, respectively. The category utility values inherent in these domains indicate that the congressional and soybean domains are learned

⁴⁷ Because COBWEB builds a classification tree in a recursive manner, characterizing behavior with respect to the top-most level also characterizes behavior with respect to lower levels.

⁴⁸ Pat Langley points out that there is a confound in the previous two experiments. For example, the intent of the last experiment was to vary the category utility score of the optimal partition across domains. However, in doing this, the number of objects in each domain was also varied. There may be some question as to whether experimental results reflect the variance in category utility or the variance in domain size. One reason that this may be a necessary confound is that it appears that the category utility score of the best partition and the number of domain objects are necessarily dependent. In particular, it does not seem possible to vary the category utility of the best partition while holding everything else (e.g., number of objects, number of attributes, number of values per attribute, category utility score of local optima) constant. However, the acknowledgement of confounding variables (necessary or not) should motivate a more extensive, but future, analysis.

easily, while the thyroid domain probably requires significantly more observations on average.

6.5 Chapter Summary

This chapter evaluated COBWEB in terms of update cost, hierarchy quality, and convergence time. Experimentation verifies that while hill climbing keeps incorporation costs down, bidirectional mobility in hierarchy space allows COBWEB to typically converge on optimal partitions over a wide range of domains. However, the effort required to converge may be distributed over a large sample of objects. Experiments show that as the quality of the optimal partition decreases, the number of objects required to ferret it out increases exponentially, and if locally optimal partitions are of sufficient quality, the global optimal may not be discovered at all.

Chapter Acknowledgements

The dimensions of this chapter were identified with the aid of Dennis Kibler. Their applicability was extended with the help of Jeff Schlimmer. Using artificial domains to delineate COBWEB's behavior was influenced by previous work by Steve Hampson and Dennis Kibler. Dennis Volper suggested the domains used to test COBWEB's ability to converge on globally (versus locally) optimal partitions.

CHAPTER 7

A Computational Account of Basic Level and Typicality Effects

7.1 Chapter Overview

Chapter 6 demonstrates that the first level of a classification tree generated by COBWEB will tend to be optimal according to category utility. While the system uses category utility to improve inference, the measure was originally motivated and validated as a predictor of the basic level [GLUC85]. Despite COBWEB's use of category utility, its classification structures and processes should not be regarded as a model of basic-level effects. Although category utility characterizes 'preferred' concepts in humans, COBWEB's explicit use of this measure does not show how this preference emerges as the result of using more primitive measures.

This chapter develops an indexing scheme and recognition procedure that accounts for basic level and typicality effects. In particular, the model offers an explanation for some target recognition (e.g., Is X a *bird*?) results. Explanations for these effects assume that object properties are directly perceivable, i.e., objects are presented to the system as attribute-value pairs. There is no claim that the model extends to the case where verbal (e.g., a word) or other symbolic cues are used to identify objects being recognized. The impact of these tasks on the model are briefly considered at the end of the chapter.

Section 2 develops a new scheme for indexing concepts and recognizing instances that involves 'extracting' certain aspects of the category utility measure. The procedure for classifying an object combines 'evidence', giving approximately the same effects as an explicit category utility calculation, but without having to

examine all categories of a level or all values of an attribute; consideration is limited to classes that are indexed by a value of the observed object.

Section 3 compares results obtained through indexing with psychological studies of basic level effects. Apparently, this is the first computational model that accounts for any basic level effect.

Section 4 extends claims of the model's psychological consistency by showing how it accounts for typicality effects. This analysis includes an explanation of experimental findings by Rosch and Mervis [Ros75B], as well as a hypothetical treatment of typicality effects in the congressional voting domain of chapter 5.

Finally, section 5 considers interactions between basic level and typicality effects. The model is consistent with interactions that have been found experimentally and predicts behaviors that have not yet been investigated. These interactions emerge as a consequence of the model's focus on concepts within a larger memory structure, rather than on isolated concepts.

In summary, this chapter presents a hierarchical indexing scheme that accounts for certain basic level and typicality effects. This chapter does not address how the indexing scheme is maintained during learning. Although object *classification* and *incorporation* are closely related in incremental systems [KOL83A], this chapter is strictly concerned with classification. Issues of indexing and memory update (i.e., learning) are addressed by COBWEB/2, a system that is described in chapter 8.

7.2 An Indexing Scheme Based On Category Utility

The classification procedure used by COBWEB recursively descends a tree along a path of 'best matching' nodes. At each level of the descent the object being classified is tentatively added to each node (i.e., class) and the resultant partition is evaluated using category utility. The node to which adding the object results

in the highest category utility score is chosen as the best host or best matching node for the object. Classification recursively proceeds through the best host. Two aspects of this procedure limit it as a model of basic level and typicality effects.

One limitation is that classification proceeds in a strict top-down fashion. If superordinate nodes exist, this procedure cannot naturally account for basic level effects. Secondly, COBWEB's classification trees cannot naturally account for typicality effects. Recall that category utility involves a summation of expressions of the form, $P(A_i = V_{ij}|N_k)^2 - P(A_i = V_{ij})^2$. Subexpressions involving some attributes will contribute more positively to the summation than will other attributes. While chapter 3 argued that typicality effects depend on attributes having varying importance for classification, an attribute's importance in the current scheme arises implicitly. A calculation for each attribute value must still be made; it cannot explain the variability of recognition time as a function of typicality. To model typicality phenomena, there must be an explicit representation of attribute importance.

To account for typicality and basic level effects, the classification scheme embodied by COBWEB is modified along two dimensions [Rosc78] of hierarchical classification. A *horizontal* dimension is concerned with the placement of objects among contrasting categories at the same tree level. While COBWEB explicitly checks each category, UNIMEM [LEBO82] and CYRUS [KOL83A] use attribute-value indices to constrain the number of possible object hosts along the horizontal dimension. Indices explicitly signify the importance of some attributes for classification. A *vertical* dimension is concerned with the placement of objects among categories at various levels of generality. Hierarchical classification usually proceeds from general to specific categories, but allowing intermediate entry points in a hierarchy results in variability along the vertical dimension; a model of human classification should allow entry points that correspond to the basic level.

7.2.1 The Horizontal Dimension: Motivating an Indexing Scheme

If an object is to be classified with respect to a number of mutually exclusive classes, it can be compared with each class using category utility. However, for purposes of classification (versus *incorporation*), operations like merging, splitting, and new node creation are not performed; classification is simply a matter of (recursively) identifying the best existing host at each tree level. Under this assumption, the category utility calculation can be significantly simplified.

Recall that category utility (3-7), $CU(\{N_1, \dots, N_n\}) =$

$$\frac{[\sum_{k=1}^n P(N_k) \sum_i \sum_j P(A_i = V_{ij} | N_k)^2] - \sum_i \sum_j P(A_i = V_{ij})^2}{n}$$

To determine the best host for a new object requires tentatively placing the object in each class and evaluating the resultant partitions by category utility. If there are n existing classes, determining a best host involves comparing n partitions, each of size n . For partitions of the same size and over the same object set, category utility (3-7) gives exactly the same ranking as

$$\sum_{k=1}^n P(N_k) \sum_i \sum_j P(A_i = V_{ij} | N_k)^2, \quad (7-1)$$

since both n and $\sum_i \sum_j P(A_i = V_{ij})^2$ are constant under these assumptions. For purposes of classification, but not incorporation, 7-1 orders partitions in exactly the same manner as 3-7. Thus, 7-1 is guaranteed to identify the same class as the best host as 3-7.

Evaluation using 7-1 is done after an object has been tentatively added to a category and appropriate counts temporarily updated. Intuitively, an object will tend to be placed in the class whose current distribution of attribute values is most reinforced by the values of the object. To place an object otherwise would lessen the predictability of a class' attribute values. This observation suggests a way of selecting a best host without evaluating the quality of the resultant

partition. Rather, the best host can be selected by independently comparing an object description with each existing class. In particular, selecting the best host for an object $O = \{A_1 = V_{1j_1}, A_2 = V_{2j_2}, \dots, A_n = V_{nj_n}\}$, can be done by finding the category, N_k , that maximizes

$$P(N_k) \sum_i P(A_i = V_{ij_i} | N_k)^2. \quad (7-2)$$

In words, the best host will tend to be the class with the greatest probabilities for attribute values of the object.

Importantly, using 7-2 is not guaranteed to identify the same class as the best host as would be identified using 7-1. However, application of 7-1 and 7-2 most always identify the same best host. As empirical evidence for this claim, consider the results of 4 COBWEB learning trials, one for each of the artificial domains of Figure 61. Object incorporation was actually done in the usual manner using function 3-7 to determine the best host, as well as to decide when to merge, split, or create a new class. However, regardless of what operator was actually applied, once the best host at the top level of the tree was selected using 3-7 (which is equivalent to 7-1 in this context), the best host was identified using 7-2. Over a sample of 75 objects from domain 1 (whose optimal partition is of least quality according to category utility), 64 of the 75 objects were classified by the same best hosts as identified by functions 3-7 and 7-2; in eleven cases these functions identified different hosts. Over samples of equal size from the remaining domains, there was never a disagreement using the two functions over which node best hosted a new object.

While 7-1 and 7-2 almost always identify the same best host, application of 7-2 is considerably cheaper. It is also more useful in developing an indexing scheme because it eliminates extraneous attribute values from the computation

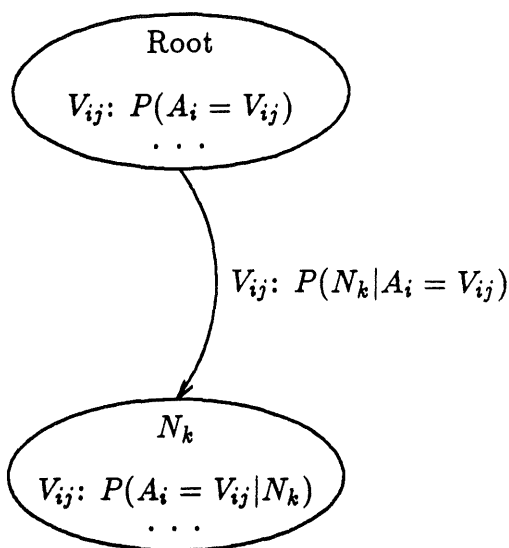


Figure 63

Sample nodes with predictability and predictiveness measures

that determines an object's class. Extraneous values are those not present in an object being recognized.

Using Bayes rule, 7-2 can be rewritten as

$$\sum_i P(A_i = V_{ij})P(N_k|A_i = V_{ij})P(A_i = V_{ij}|N_k). \quad (7-3)$$

Although equal to 7-2, 7-3 suggests an appealing way of mapping component probabilities onto a classification tree structure. In particular, chapter 3 pointed out that $P(N_k|A_i = V_{ij})$ or the cue validity of V_{ij} is a measure of predictiveness, while $P(A_i = V_{ij}|N_k)$ or category validity is a measure of a value's predictability at N_k . Similarly, $P(A_i = V_{ij}) = P(A_i = V_{ij}|\text{Root})$, is a measure of the predictability of V_{ij} at the root of a classification tree. Figure 63 shows these probabilities distributed over two classification tree nodes.

Indices can be used to direct classification. For example, COBWEB compares an object against each node among a set of siblings. In contrast, UNIMEM

[LEBO86, LEBO82] and CYRUS [KOL83A, KOL83B] use a two-step recognition process. Attribute value indices are used to identify nodes that *may* classify the object. Thus, indexing is a relatively cheap way of filtering out nodes that are not similar to the new object. Nodes deemed relevant through indexing are then evaluated more completely with respect to the object.

A method of distributing category utility information over a classification tree has been sketched. Individual attribute values index subordinate concepts. A comprehensive match between an object and possible hosts is preceded by an indexing stage that initially identifies relevant concepts. Indexing is guided by cue validities on arcs. The information needed for a complete match (i.e., computing category utility) is contained at nodes, as well as arcs, of a tree. In a manner similar to that employed by UNIMEM and CYRUS, this procedure can identify a best host for an incoming object. However, in UNIMEM and CYRUS, indices are restricted to connecting nodes to their immediate children. Under this assumption, indexing aids classification only along the horizontal dimension. In fact, systems that do hierarchical classification move strictly top – down; choices of classification along the vertical (or generality) dimension are not considered. Before fleshing out a two-step classification procedure that has been inspired by category utility, implications of the vertical dimension are considered.

7.2.2 The Vertical Dimension: Placing Indices

If a value is relatively unique to members of a subordinate node, it will indicate the subclass roughly to the same degree that any superordinate node is indicated. For example, in a hierarchy containing *animal*, *vertebrate*, *mammal*, *dog*, *collie*, having hair is unique to *mammals*; $P(\text{mammals}|\text{have-hair}) = P(\text{vertebrates}|\text{have-hair}) = 1.0$. Rather than directing an index for have-hair from *animals* to *vertebrates*, an index can bypass *vertebrates* and go directly to *mammals*. *Mammals* is the node where have-hair is necessary and sufficient for membership.

More generally, the rule for index placement can be relaxed by directing indices at nodes that maximize collocation for the corresponding attribute value. Collocation is a tradeoff of value predictability (i.e., $P(A_i = V_{ij}|N_k)$) and predictiveness (i.e., $P(N_k|A_i = V_{ij})$), which generalize logical necessity and sufficiency, respectively.

As discussed in chapter 5, values that maximize collocation at a node are considered normative of the node. These values approximate conditional independence from other values. Moreover, collocation maximizing indices will tend to be directed at subordinate (i.e., specific) nodes. Taken together, conditional independence and greater specificity tend to insure that value indices are directed at nodes from which the most can be predicted from simply knowing the value.

Allowing indices to skip levels introduces variability along Rosch's vertical dimension of classification. A tree that is indexed in this way is partially shown in Figure 64 as an example. This indexed tree corresponds to the unindexed tree over animals given in Figure 36 of chapter 4. Dashed lines indicate parent - child relationships. Although node probabilities (predictability) are not shown for space reasons, predictiveness scores for certain arc labeling attribute values are shown. Each arc corresponds to an attribute value whose collocation is maximized at the node where the arc terminates. For example, 'HeartChambers = four' predicts the 'mammals/bird' node with probability 1.0 and its predictability at this node is 1.0 as well. The collocation of 'HeartChambers = four' is $1.0 \times 1.0 = 1.0$ which is the maximal score for this attribute value. Thus, an arc for 'HeartChambers = four' is directed from the root to the 'mammals/bird' node. Recalling discussion from chapter 5, arcs are directed at nodes where the applicable node is normative. While space only permits showing arcs from the root of the tree, the indexing procedure is applied recursively; each node is treated as the root of its own subtree.

Recognition using this scheme is a two-step process that begins at the root of a classification tree. Possibly relevant nodes are activated by indexing. A

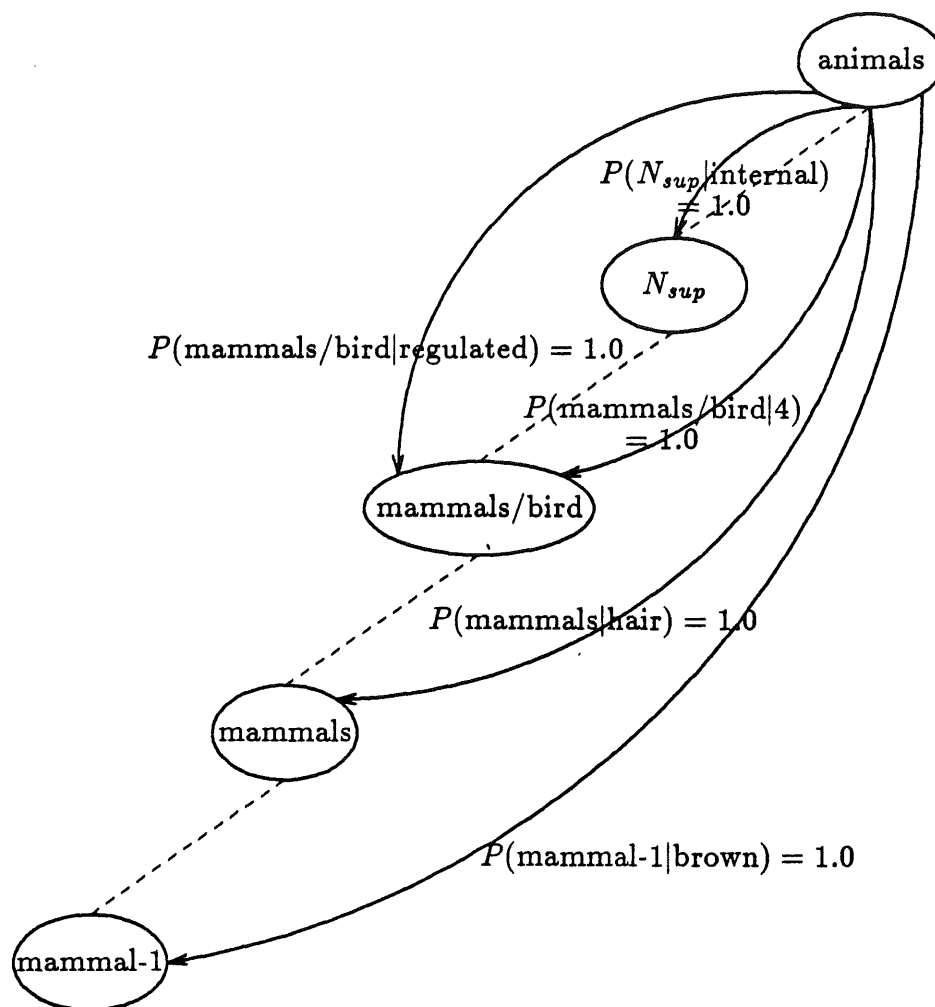


Figure 64

A partially indexed tree of animals

more complete evaluation is then made to determine where an object resides. In particular, the following scheme is used to classify an object, $O = \{A_1 = V_{1j_1}, A_2 = V_{2j_2}, \dots, A_m = V_{mj_m}\}$:

INDEX: Return a set of relevant nodes that might classify an object, O , by:

- a) Compute the sum, $\sum_i P(N_k | A_i = V_{ij_i})$, for all nodes, N_k , over all V_{ij_i} that index N_k and are present in O .
- b) Identify the node, N_{max} , with the maximum sum (i.e., the most highly predicted node).
- c) Identify all nodes that share indexing values with N_{max} . These nodes, together with N_{max} , are kept for further evaluation.

Activated nodes that share one or more indexing values with N_{max} cannot be ancestors or descendents of N_{max} , since only one node in an ancestral line can maximize collocation for a given value. Thus, the INDEXing stage returns a set of nonancestral related nodes. These nodes may then be more carefully evaluated by the following procedure.

EVALUATE: The set of nodes that were kept following activation are evaluated by computing function 7-3 over all object values. The node that maximizes 7-3 is assumed to classify the object.

An example of this recognition procedure is illustrated using the partially indexed tree of Figure 64. Consider the problem of classifying 'mammal-1' from Table 5 of chapter 4. This object is described by {BodyCover = hair, HeartChambers = four, BodyTemp = regulated, Fertilization = internal, Color = brown}. Each of the nodes (except the root) shown in Figure 64 are indexed by at least one value of this object. Moreover, because each value predicts one of the given nodes with probability 1.0, these are the only nodes indexed by a value of this object. Of these nodes, 'mammals/bird' has a total predictiveness score of $1.0 + 1.0 = 2.0$. This is the maximally predicted node and is identified as N_{max} . Furthermore, no other indexed node shares an indexing value with 'mammals/bird'. Therefore, this node is the only one retained for evaluation and is thus chosen to initially classify 'mammal-1'.

After a best host has been determined, classification recursively proceeds to deeper levels, eventually terminating at a leaf. Invariably (with respect to the example above and the following experiments), N_{max} maximizes 7-3 from among

the activated nodes. In many cases, this happens because N_{max} is the only node passed on for evaluation. However, regardless of the number of nodes passed on by indexing, there are good reasons for the great favoritism towards N_{max} . Because indices are directed at nodes that maximize collocation, $P(A_i = V_{ij}|N_{max})$ tends to be quite high for indexing values. Therefore, the total predictiveness score of N_{max} equals

$$\sum_i P(N_{max}|A_i = V_{ij}) = \sum_i P(N_{max}|A_i = V_{ij}) \times 1.0,$$

which approximates

$$\sum_i P(N_{max}|A_i = V_{ij})P(A_i = V_{ij}|N_{max}).$$

Moreover, because $P(A_i = V_{ij})$ for each value is constant across all activated nodes, it will tend to have little impact in selecting between N_{max} and its competitors. In chapter 8, the two-step classification procedure is followed for tree *update*, but in the remainder of this chapter, no mention of the EVALUATE stage will be made.

7.3 Basic Level Effects

Using attribute value indices that are directed at collocation maximizing nodes tends to result in objects being first recognized with respect to the basic level. The reason for this behavior is intuitively simple: category utility predicts that the basic level of a hierarchy maximizes a tradeoff between the ability of attribute values to discriminate categories (using $P(N_k|A_i = V_{ij})$) and the ability of values to characterize categories (using $P(A_i = V_{ij}|N_k)$). Specifically, a basic-level node tends to be that node among its descendents and ancestors that maximizes collocation for the most frequently observed attribute values. Since attribute-value indices are directed to those node(s) which maximize collocation for individual

values, indices collectively tend to maximally predict basic level nodes. In this section, this intuition is validated using two experimental studies.

7.3.1 Explanation of the Murphy and Smith Data

An experimental study was conducted by Murphy and Smith [MURP82]. They presented subjects with 16 line drawings of simple, abstract tools, including hammers, bricks, pizza cutters, and knives. Tools were arranged into a hierarchy of five levels. Subjects were trained to recognize a tool at each of the four levels of abstraction (e.g., a large, clawed, hammer could be identified as such, or as simply a clawed hammer, hammer, or pounder) using fictional names provided by the experimenters. Following the training phase, Murphy and Smith employed a target recognition task to determine the level treated by subjects as basic. Subjects behaviorally identified the intermediate level containing nodes hammer, brick, knife, and pizza cutter as basic, since recognition was consistently verified more quickly with respect to these categories.

In order to test the validity of the indexing scheme, the pictorial representations used by Murphy and Smith must be converted to an attribute-value representation. As pointed out by Gluck and Corter [GLUC85], the nature of such a transformation is the focus of a good deal of research in vision and pattern-recognition. However, the drawings used in these experiments were varied along four perceptual dimensions: the shape of the tool handle, the shaft, and the head, as well as the overall size of the drawing. Secondly, subjects were told that the superordinate distinction implicit in the tool classification tree was in terms of function: cutting tools (knives and pizza cutters) vs. pounding tools (hammers vs. bricks). These considerations lead to an encoding in terms of five attributes, which is shown in Table 21. An encoded version of the leftmost portion of the tree presented to human subjects is shown in Figure 65. Leaves give an abbreviated each object description as a string that is ordered as *Function, Handle, Shaft,*

Category (Node)		Attributes						
Superordinate	Basic	Function	Handle	Shaft	Head	Size		
Pounder	Hammer	Pounding	2	2	2	0		
		Pounding	2	2	2	1		
		Pounding	2	2	1	0		
		Pounding	2	2	1	1		
	Brick	Pounding	0	3	4	0		
		Pounding	0	3	4	1		
		Pounding	1	3	4	0		
		Pounding	1	3	4	1		
		Cutter	Knife	Cutting	3	4	2	0
				Cutting	3	4	2	1
Cutting	3			4	3	0		
Cutting	3			4	3	1		
Pizza C.	Cutting		4	0	5	0		
	Cutting		4	0	5	1		
	Cutting		4	1	5	0		
	Cutting		4	1	5	1		

Table 21

An attribute - value encoding of the Murphy and Smith tools

Head, Size. For example, 'P2220' stands for {Function = Pounding, Handle = 2, Shaft = 2, Head = 2, Size = 0}. Internal nodes are listed by the fictional class names provided by the experimenters and used by subjects to verbalize object identifications. This encoding is identical to that presented by Gluck and Corter, except that Table 21 lists function as an attribute, where Gluck and Corter did not. In the computer experiments to follow, it turns out that adding function does not change the node that is first used to classify an object.

Using the encoding of Table 21, Figure 66 illustrates the indexing scheme imposed on the classification tree of simple tools. Each node lists those attribute values that have collocation maximized at the node. To simplify discussion, indices are shown only for the leftmost nodes of the tree, but this path is representative of all others. Associated with each index is the probability of the indicated node given

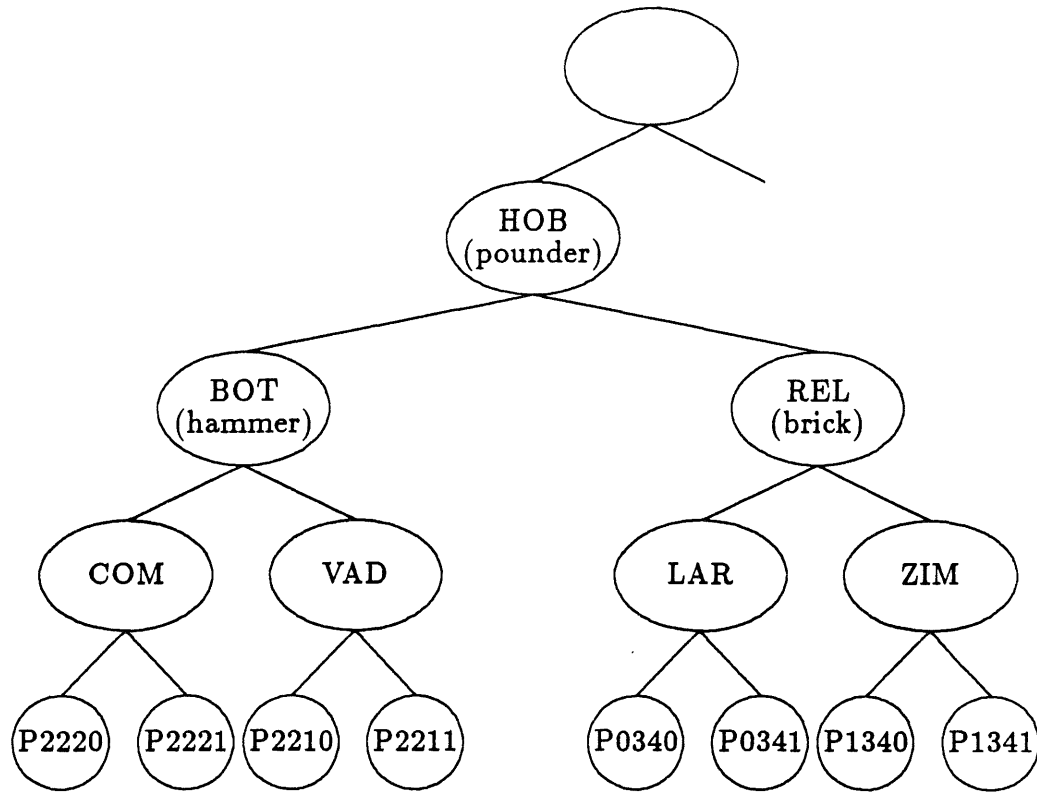


Figure 65

Attribute-value encoding of the Murphy and Smith tree

the indexing attribute value. Similarly, associated with nodes are the probabilities of attribute values conditioned on category membership.

The recognition procedure predicts that all tools are first recognized with respect to basic nodes as indicated by human subjects. For example, when a small, clawed, hammer is presented (encoded as function = 0, handle = 2, shaft = 2, head = 2, size = 0), the predictiveness scores computed during the first stage of recognition are as follows:

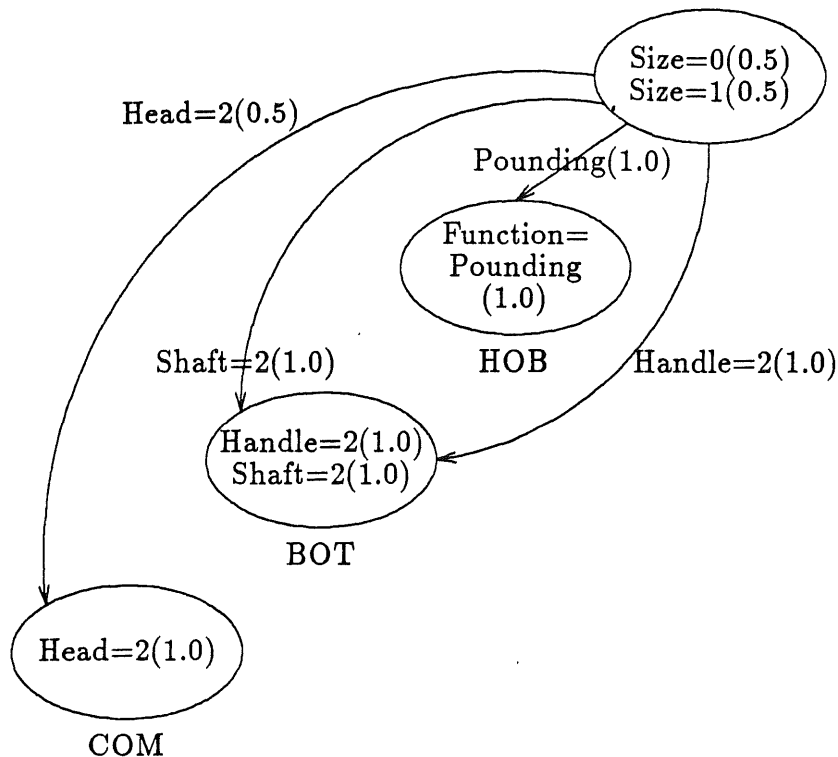


Figure 66

Indexing imposed on the tree of Murphy and Smith experiment

Node	Predictiveness Score
Pounder	$P(\text{Pounder} \text{Function} = \text{Pounding}) = 1.0$
Hammer	$P(\text{Hammer} \text{Shaft} = 2) + P(\text{Hammer} \text{Handle} = 2)$ $= 1.0 + 1.0 = 2.0$
Clawed Hammer	$P(\text{Clawed Hammer} \text{Head} = 2) = 0.5$

The node with the highest predictiveness score is Hammer with a score of 2.0. This node is retained for the second stage of processing. Moreover, since no other nodes with indexing values that intersect with Hammer are activated, Hammer is the only one retained. This example indicates that an object description corresponding to a small, clawed, hammer is first recognized with respect to the basic node as indicated by human subjects. Similar verification of the basic level phenomena can be made for all objects (tools) of this domain.

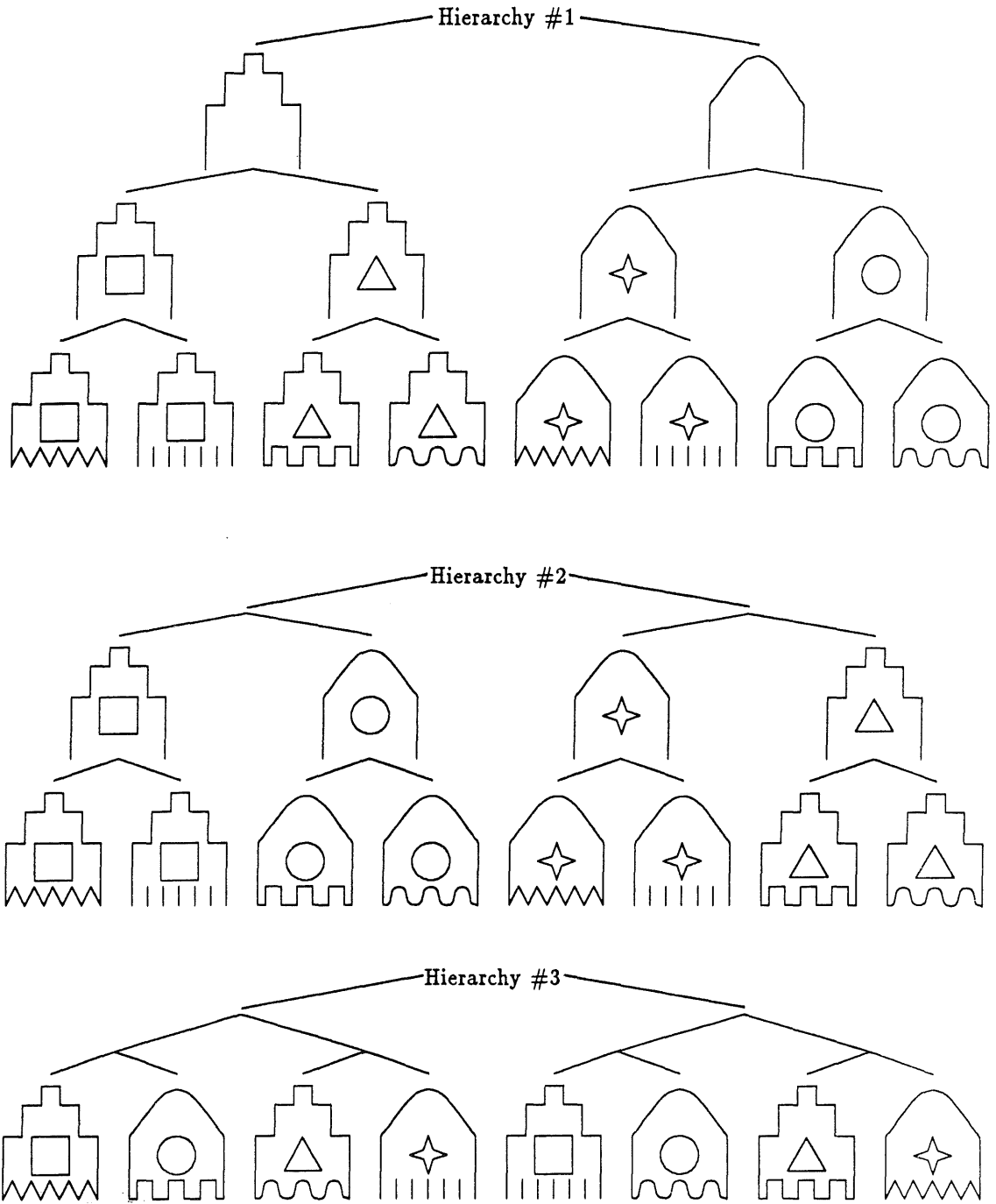


Figure 67

Hierarchies of objects defined by outer, inner, and bottom shape

7.3.2 Explanation of the Hoffman and Ziessler Data

The recognition/indexing scheme gives results that are consistent with a second set of experiments conducted by Hoffman and Ziessler [HOFF83]. Subjects were trained to recognize eight different artificial objects. Objects were arranged into classification trees, similar to those used by Murphy and Smith. Unlike the earlier experiment however, Hoffman and Ziessler arranged the objects into three different tree structures and trained a different set of subjects on each tree. The trees are given in chapter 3, but are presented again for convenience in Figure 67.

The goal of arranging objects into three hierarchies was to impose a different basic level in each case. Target recognition tasks verified that subjects identified a different level as basic for the three cases. Objects in this domain were characterized in terms of three attributes, corresponding to the shape of an object's outer perimeter, the shape of a subobject residing inside the outer object, and the shape of the bottom of the outer object. The attribute value encoding for each tree, along with the basic level is presented in Table 22.

The indexing/recognition scheme unambiguously predicts recognition with respect to the basic level in the majority of cases, but in one case there is a tie. Consider the indexing scheme imposed onto the leftmost portion of tree 2 in encoded form shown in Figure 68. The collocation of 'Inside = 0' is maximized at the node pointed at by the appropriate arc. Note that level one nodes are not indexed by any values since no value maximizes collocation at these nodes. The figure only shows indices that emanate from the root. As pointed out in previous examples, indices also emanate from lower level nodes, thus allowing an to be recursively classified with respect to lower level nodes.

Subjects behaviorally identified level two of this tree (where the root is at level 0) as basic. When presented with the object, (Shape = 0, Inside = 0, Bottom = 0), two nodes (one is not shown) at level two are indexed by Shape = 0 with

Tree 1					
(Basic) Superordinate	Category (Node)		Attribute		
	Middle	Instance	Shape	Inside	Bottom
High 1	Mid 1	1	0	0	0
		2	0	0	1
	Mid 2	3	0	1	2
		4	0	1	3
High 2	Mid 3	5	1	2	0
		6	1	2	1
	Mid 4	7	1	3	2
		8	1	3	3

Tree 2					
(Basic) Superordinate	Category (Node)		Attribute		
	Middle	Instance	Shape	Inside	Bottom
High 1	Mid 1	1	0	0	0
		2	0	0	1
	Mid 2	3	1	3	2
		4	1	3	3
High 2	Mid 3	5	1	2	0
		6	1	2	1
	Mid 4	7	0	1	2
		8	0	1	3

Tree 3					
(Basic) Superordinate	Category (Node)		Attribute		
	Middle	Instance	Shape	Inside	Bottom
High 1	Mid 1	1	0	0	0
		2	1	3	2
	Mid 2	3	0	1	3
		4	1	2	1
High 2	Mid 3	5	1	2	0
		6	0	1	2
	Mid 4	7	1	3	3
		8	0	0	1

Table 22

Attribute value encodings of trees used by Hoffman and Ziessler

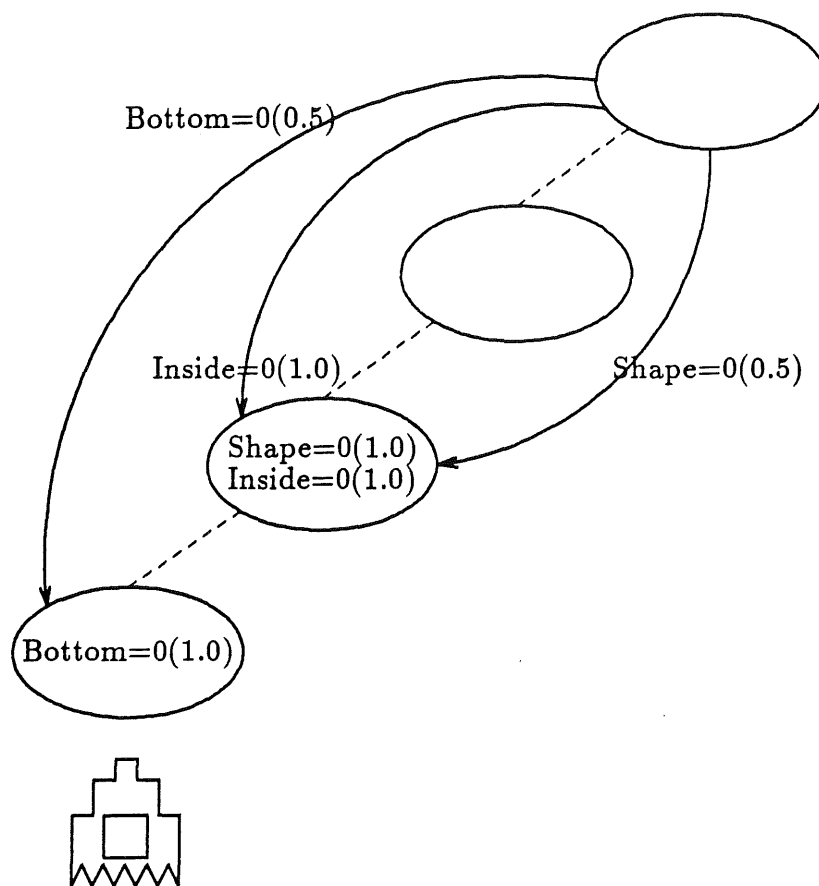


Figure 68

 Indexing imposed on tree 2 of Hoffman and Ziessler Experiments

predictiveness 0.5 each, while $\text{Inside} = 0$ indexes one of these with a predictiveness of 1.0. Two nodes at level three are indexed by $\text{Bottom} = 0$ with a predictive score of 0.5 each. In contrast, the leftmost level two node has a total predictiveness score of 1.5 which is greater than any other indexed node. By a similar process, all objects are recognized with respect to the basic level when classified using tree 2. Moreover, all objects are classified with respect to the bottommost level of tree 3, which is basic in this case.

While objects are classified unambiguously with respect to the appropriate basic level in the case of trees 2 and 3, some ambiguity arises in the case of tree 1. In this case, nodes of the top and middle levels are predicted equally. This occurs

because nodes at each level maximize collocation for a single attribute value, and this value is perfectly predictive of the node. For example, when presented with the object (Shape = 0, Inside = 0, Bottom = 0), the appropriate level 1 (basic level) node has a predictive score of 1.0, since Shape = 0 predicts this node with probability 1.0. Likewise, the level 2 node indexed by Inside = 0 has a predictive score of 1.0. Since both nodes are predicted by the same amount, the selection of which to retain for the second phase of processing is arbitrary.

There are several possible explanations for the preference by human subjects for the top level of tree 1. The first, which cannot be addressed by the current scheme, is that outer shape may be more salient to subjects than inside shape, where salience is a measure of an attribute's 'noticability', and not tied to the probability of observation used in chapter 4. A second explanation concerns the use of the base rate probability of attribute values. In the current scheme a sum of probabilities of the form $P(C_k|A_i = V_{ij})$ is used to compute predictiveness. An alternative model could include base rate probability in the computation of predictiveness, i.e., a sum of $P(A_i = V_{ij})P(C_k|A_i = V_{ij})$. In the case of tree 1, the top level would be unambiguously selected to classify an object with Shape = 0 and Inside = 0, since $P(\text{Shape} = 0) = 0.5 > P(\text{Inside} = 0) = 0.25$. However, using this scheme would simultaneously result in a tie between levels 1 and 2 of tree 2. Thus, each scheme results in a tie with respect to one of the Hoffman and Ziessler trees. The selection of the current computation of total predictiveness was made because it corresponds exactly with the psychological measure of total cue validity discussed in Chapter 3. Second, the product $P(A_i = V_{ij})P(C_k|A_i = V_{ij}) = P(A_i = V_{ij} \wedge C_k)$ has less intuitive and mathematical appeal as a measure of predictiveness than $P(C_k|A_i = V_{ij})$.

7.3.3 Summary

The indexing scheme and recognition procedure classify objects with respect to the basic level as identified by human subjects in two experimental studies. Of course, there are many psychological studies of basic level effects that have not been examined in this section. Of these, many study basic level effects in natural domains (e.g., animals). In these cases, a rigorous comparison between the model's and human performance is difficult, since there is no way of controlling the features that human subjects use to encode observations. Furthermore, this section only presents evidence for the consistency of the model with results from target recognition tasks. There has been no attempt to account for many other studies of basic level effects [MERV81, ROSC78]. Nonetheless, apparently this is the first computational model intended to account for any basic level effect, although it borrows heavily from work by Gluck and Corter [GLUC85], as well as Jones [JONE83].

7.4 Typicality Effects

The indexing scheme of this chapter accounts for basic level effects in two experimental studies and appears to be consistent with human preferences along Rosch's vertical dimension of classification. However, humans also exhibit preferences along a horizontal dimension. These preferences are evidenced in typicality studies. This section demonstrates that indexing accounts for typicality effects in three domains, two of which are artificial domains used in experiments by Rosch and Mervis [ROS75B], as well as the congressional domain of chapter 5.

7.4.1 Explanation of the Rosch and Mervis Data

Several studies by Rosch and Mervis [ROS75B] were used to elicit typicality effects from human subjects. These studies tested the hypotheses that an instance's typicality was dependent on the instance's 'family resemblance' to other class

Category	Letter String	Within-Category Overlap	Between-Category Overlap
A	JXPHM	15 (low)	0
	QBLFS	15	0
	XPHMQ	19 (medium)	0
	MQBLF	19	0
	PHMQB	21 (high)	0
	HMQBL	21	0
B	CTRVG		
	TRVGZ		
	RVGZK		
	VGZKD		
	GZKDW		
	ZKDWN		

Table 23

Categories used to test the effect of intra-class similarity on typicality

members. In particular, they tested how behavioral indicators of typicality (e.g., response time) varied with an object's similarity to members of the same class and a contrasting class.

7.4.1.1 The Effect of Intra-category Similarity

Rosch and Mervis used nonsense strings to study typicality effects [Ros75B]. 'Objects' of this type allowed them to easily manipulate within- and between-category similarity. The effect of varying intra- (within-) category similarity on typicality was studied using the categories of Table 23.⁴⁹ Category A was constructed so that each member shared properties with other members of category

⁴⁹ Rosch and Mervis actually used ten different sets of data, each composed of a different combination of letters and numbers. However, the following analysis assumes that because each set was 'identically structured', the conclusions reached for analysis of one data set (the only one published) are tentatively extendable to all data sets. This seems to be reasonable assumption because while Rosch and Mervis presumably used different symbol combinations to rule out the possibility of certain salience effects, the computer model is oblivious to such preferences.

A to varying extents, but did not overlap at all with any member of category B. For example, there are 19 cases of the letters of 'XPHMQ' in category A (i.e., 2 'X's + 3 'P's + 4 'H's + 5 'M's + 5 'Q's) and no cases of a letter from this string in any member of category B. Thus, between-category similarity was held constant, while within-category similarity was varied. For purposes of analysis, Rosch and Mervis grouped category A members into three subsets: the two strings with *low* (i.e., 15), *medium* (i.e., 19), and *high* (i.e., 21) overlap with other category A members.

Subjects were taught to distinguish categories A and B. After learning, subjects participated in a target recognition task; they verified category membership for each learned string (e.g., Is 'JXPHM' a member of Category A?). In addition to target recognition times, averaged results of subject's subjective judgements of instance typicality (using a 6 point scale) and the average number of errors in classification were collected. This study supported the hypothesis that category A members sharing more symbols with members of the same category tend to be recognized more quickly, judged more typical, and are less frequently misidentified.

An explanation of these effects can be constructed using the indexing scheme based on category utility. However, this explanation presumes that indexing is imposed on an existing classification tree. For purposes of testing indexing's consistency with human classification, in this and future experiments, two trees are tested. One tree forced objects of the same externally defined categories (i.e., A and B) to be classified under the same node of the tree. A second tree is built by COBWEB and groups objects based on attribute-value similarity, irrespective of their externally defined classes. In Rosch and Mervis' experiments categories A and B were taught by a process of learning from examples, perhaps biasing subjects to segregate members of A and B. However, it is possible, if not probable, that similarities along other attribute values impact object grouping. Rather than testing the indexing scheme with respect to a single tree, the two trees

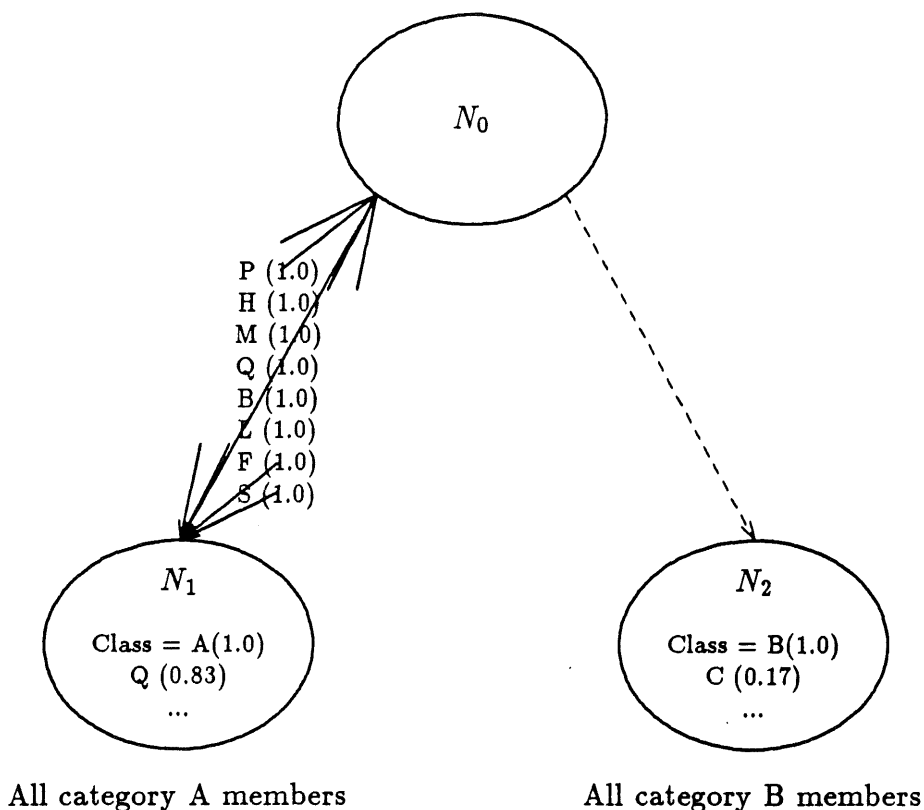


Figure 69

Partially indexed tree over strings of Table 23.

represent extremes of possible encodings. An assumption of upcoming analysis is that if classification using these trees is consistent with human classification, tree encodings of intermediate forms will be as well.⁵⁰

Consider the tree of Figure 69, which covers the strings of Table 23. This tree was formed by manually segregating members of categories A and B. The tree is 'flat', only decomposing the object set to one level deep. Indexing is shown for a subset of the observed letters; these are sufficient for demonstrating recognition for a string of low ('QBLFS') and high ('PHMQB') within-category overlap. In placing indices, each string was encoded in terms of attribute - value pairs; each

⁵⁰ One tree of intermediate form might result from running COBWEB on data in which class designation was included in object descriptions, but is simply treated as another attribute. However, trees so generated are not investigated here.

letter (e.g., 'Q') was considered an attribute whose value was 'present' (e.g., 'Q = present'). Because no value is shared between category A and B members, each value is perfectly predictive of the node covering a single class. For example, $P(N_1|P) = 1.0$ because N_1 contains all and only members of category A. For this reason also, once an object is classified with respect to N_1 , it can be asserted as a member of category A.

To demonstrate classification using the tree of Figure 69 consider 'PHMQB'. All indices for letters of this string are individually predictive of N_1 with probability 1.0 (i.e., $P(N_1|P=\text{present}) = 1.0$). Considering all indices, N_1 is indicated by a total predictiveness score of 5.0; N_1 is the only node indicated. Since only category A members are stored at N_1 (i.e., $P(\text{Category}=A|N_1) = 1.0$), 'PHMQB' can be unambiguously asserted to be a member of category A. In fact, because of the flatness of the tree of Figure 69 and the lack of overlap between letters found in category A and B members, all members of category A indicate N_1 with a total predictiveness of 5.0. This can be verified for 'QBLFS', whose values are also shown as indices in Figure 69. If recognition time is assumed to be only a function of total predictiveness, the current tree offers no explanation of response time differences between instances with varying intra-class similarity.

A second tree over the data of Table 23 is shown in Figure 70. This tree was constructed by COBWEB. Because of the lack of overlap between category A and B members, this tree also partitions category A and B members under two nodes. Category A instances are all classified under N_1 . However, unlike the manually segregated tree, COBWEB's tree decomposes the data set all the way to individual objects at tree leaves. Two of N_1 's descendents, N_2 and N_4 , are shown. Indices are shown for the letters of strings 'QBLFS' and 'PHMQB'. However, because the tree is deeper than one level, indices are spread across several levels. As examples, the letter 'Q' is common to most category A members and its collocation score

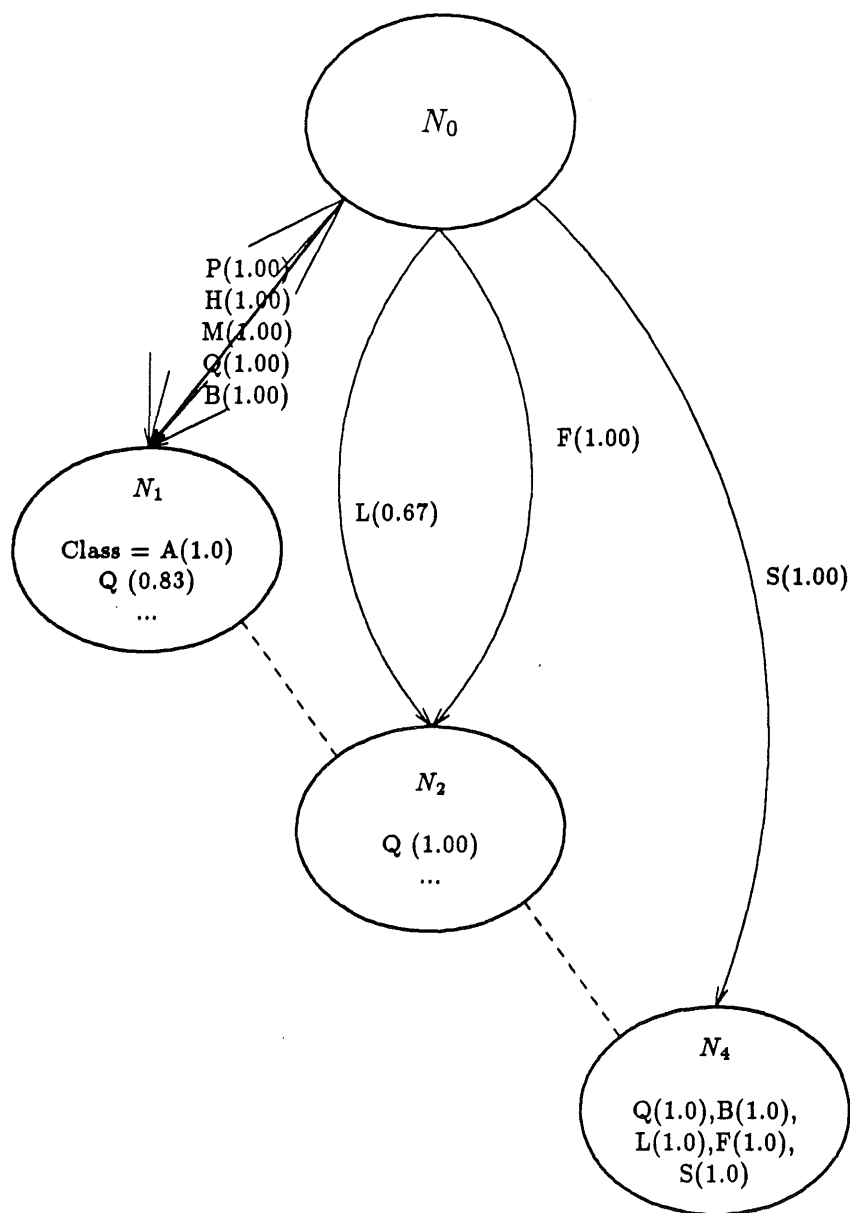


Figure 70

Partially indexed (COBWEB) tree over strings of Table 23 .

is maximized at N_1 , while 'S' is unique to 'QBLFS' and its collocation score is maximized at the leaf N_4 .

For 'PHMQB', indexing indicates N_1 with a total predictiveness of 5.0. This is the same score obtained with the first tree. However, classification of 'QBLFS'

Class A instance	Intra-category similarity	Human time (ave.)	Total predict score to N_1	Average model time (ave.)
JXPHM QBLFS	low	692 ms	2.5	0.40 units
XPHMQ MQBLF	medium	617	3.5	0.29
PHMQB HMQBL	high	560	4.5	0.22

Table 24

Human and computer model response times (1)

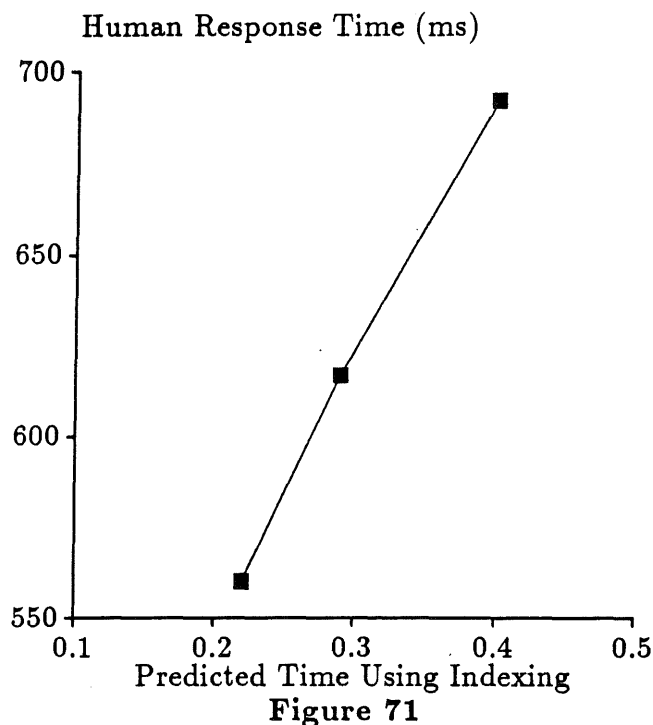
yields a different result. Because collocation for some letters are maximized at nodes subordinate to N_1 , not all indices corresponding to the letters of 'QBLFS' are directed at N_1 . For example, indices for 'L' and 'F' indicate N_2 with probabilities 0.67 and 1.0, respectively. An index for 'S' indicates N_4 with probability 1.0. Indices for 'QBLFS' are spread over nodes at three levels, thus diffusing the total predictiveness with respect to each node. The total predictiveness scores for 'QBLFS' and each indexed node are:

$$\text{Total predictiveness of } N_1 = P(N_1|Q) + P(N_1|B) = 2.0$$

$$\text{Total predictiveness of } N_2 = P(N_2|L) + P(N_2|F) = 1.67$$

$$\text{Total predictiveness of } N_4 = P(N_4|S) = 1.0$$

Node N_1 is indicated with the greatest total predictiveness score, 2.0. Additionally, there are no other nodes indexed with overlapping letters and so N_1 is chosen to classify 'QBLFS'.



Human and computer simulated response times

Indexing the tree produced by COBWEB explains the variation in response time resulting from differences in intra-class similarity: collocation for attribute values that are shared by many class members will tend to be maximized at the same node and instances that possess many of these frequently occurring values will more strongly suggest a single central node. Values that are not universally shared by class members may be directed to subordinate nodes and diffuse the predictiveness of an instance over several tree levels. This explanation assumes that the more an instance predicts a node, the less time required to reach the node. That is, response time is inversely proportional to total predictiveness. The results of Table 24 are consistent with this conclusion.

More specifically, the model can be used to predict response time by assuming that total predictiveness is a measure of the rate at which activation proceeds from one node to another. A further assumption is that the distance between any

two nodes is equal; for convenience, this distance will be 1.0. A straight forward measure of the time it takes to reach a subordinate node from the root is given by

$$\text{time} = \frac{\text{distance}}{\text{rate}} = \frac{1.0}{\text{total predictiveness}}$$

This equation allows a comparison between human response time and the response time predicted using category utility based indexing. For example, the time to reach N_1 when observing 'QBLFS' is $\frac{1}{2.0} = 0.5$ units, while the time for 'PHMQB' is $\frac{1}{5.0} = 0.2$. Reaching N_1 permits an assertion that the instance is a member of category A. Table 24 also shows the average predicted response times for instances with low, medium, and high intra-class similarity. Figure 71 graphically shows that predicted response times order the instances of category A in the same manner as human subjects. The use of a simulated response time is for conceptual convenience; there is no strong commitment to this particular measure of time. However, this measurement of response time represents one possible instantiation of a firmer commitment: response time is inversely proportional to total predictiveness.

7.4.1.2 The Effect of Inter-category Similarity

In a second experiment, Rosch and Mervis [Ros75B] tested the impact of varying inter-class similarity on typicality. Table 25 shows the strings used in this study. Each member of category A shared an equal number of symbols with other members of category A, but differed in the number of letters shared with category B. For example, the symbols of 'HPNSJ' occurred a total of 12 times over category A instances (i.e., 3 'H's + 3 'P's + 3 'N's + 1 'S' + 2 'J's) and two symbols appear in category B (i.e., 'S' and 'J'). Thus, within-category similarity was held constant, while between-category similarity was varied. In a manner similar to the previous experiment, category A members were grouped into three subsets: the two strings with *low*, *medium*, and *high* overlap with category B members.

Category	Letter String	Within-Category Overlap	Between-Category Overlap
A	HPNWD	12	0 (low)
	HPC6B	12	1
	HPNSJ	12	2 (medium)
	4KC6D	12	3
	GKNTJ	12	4 (high)
	4KCTG	12	5
B	8SJKT		
	8SJ3G		
	9UJCG		
	4UZC9		
	4UZRT		
	MSZR5		

Table 25

Categories used to test the effect of inter-class similarity on typicality

Subjects were taught to distinguish categories A and B. After learning, subjects verified category membership for each learned string (e.g., Is 'HPNWD' a member of Category A?). Table 25 shows the results of this experiment. Response times have been averaged for instance pairs corresponding to those with low, intermediate, and high inter-class overlap. In addition to target recognition times, Rosch and Mervis recorded subjective judgements of instance typicality and the average number of classification errors. Rosch and Mervis verified that category A members sharing fewer symbols with members of the contrasting category tended to be recognized more quickly, judged more typical, and led to fewer identification errors.

Indexing based on category utility accounts for the typicality findings of this experiment. Consider indexing applied to the tree of Figure 72. This tree corresponds to the case where category A and B members are manually partitioned under the same nodes, N_1 and N_2 . Collocation for 6 symbols (i.e., 'S', 'J', 'T', 'G',

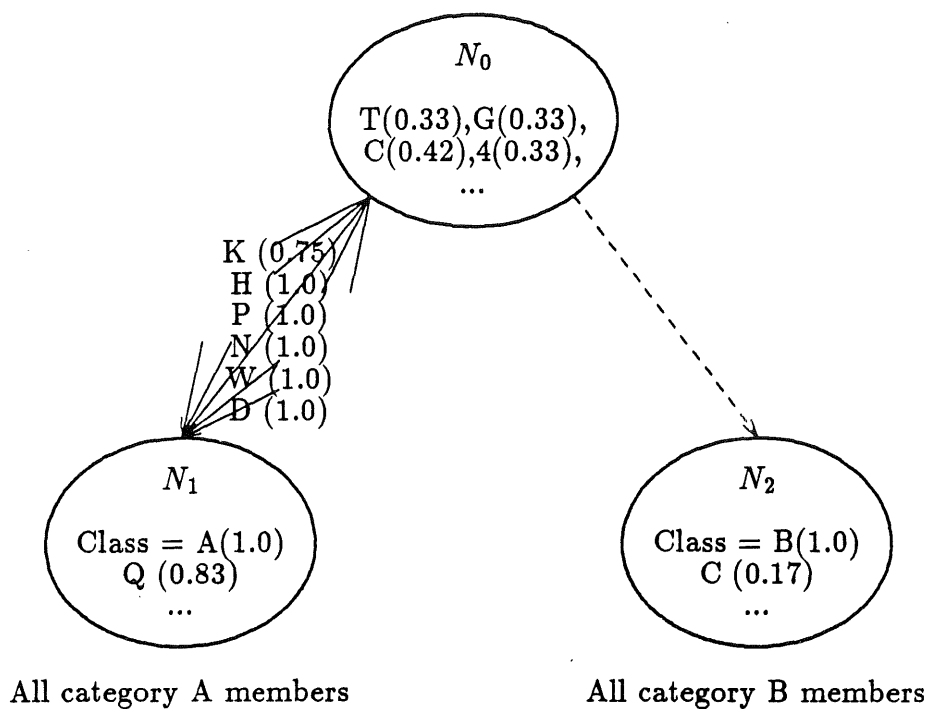


Figure 72

Partially indexed (manual) tree over strings of Table 25.

Class A instance	Inter-category similarity	Human time (ave.)	Total predict score to N_1	Average model time (ave.)
4KCTG	high	1125ms	1.6	0.63 units
GKNTJ				
4KC6D	medium	986	2.9	0.35
HPNSJ	low	909	4.5	0.22
HPC6B				
HPNWD				

Table 26

Human and computer model response times (2)

'C', '4') are maximized at the root; no indices for these symbols emanate from the root. The presence of seven norms at the root signifies that many symbols are not predictive of either category A or B, rather they are distributed about evenly

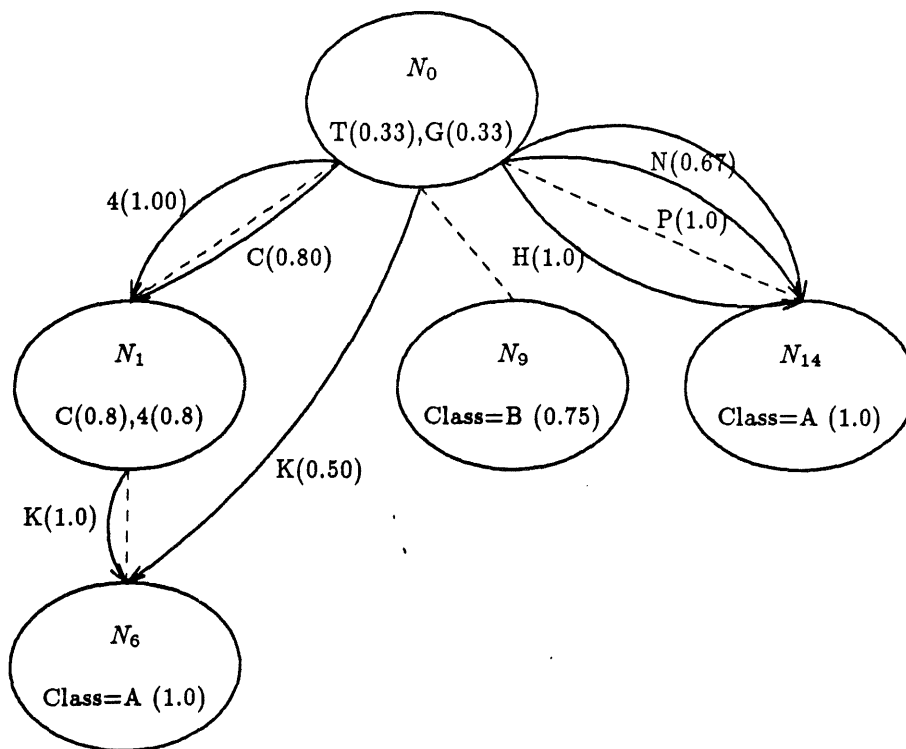


Figure 73

Partially indexed (COBWEB) tree over strings of Table 25.

in each category. Using this tree, category A members that significantly overlap with category B will tend to have values that are norms at the root; they do not contribute to total predictiveness. For example, '4KCTG' has four values that are root norms (i.e., '4', 'C', 'T', 'G'). Only 'K' is predictive of N_1 . This string's total predictiveness equals the predictiveness of 'K', 0.75. On the other hand, 'HPNWD' does not share any symbols with category B and has a total predictiveness score of 5.0. Simulated response times can be calculated as discussed earlier. Human and computer model response times for strings with low, medium, and high inter-class overlap are shown in Table 26. In the case of the tree of Figure 72, predicted response times rank strings in the same manner as human subjects.

In addition to the previous tree, a tree constructed by COBWEB also accounts for typicality differences that result from changes in inter-category similarity. Figure 73 shows part of a tree that was built by COBWEB over the strings of Table 25. The COBWEB tree does not necessarily place objects of the same category (i.e., A or B) under the same node. Rather, it groups objects that are most similar according to category utility. For example, under N_1 are two strings from category A and three from category B. Not coincidentally, the two strings from category A (i.e., '4KCTG' and '4KC6D') share a majority of symbols (i.e., five and three, respectively) with category B members. The third category A member that shares a majority of symbols with category B is 'GKNTJ'; this string is stored under N_9 along with the remaining three members of category B. On the other hand, the three strings of category A with least category B overlap reside together and alone at N_{14} . The segregation of these three instances will make them easier to recognize as members of category A.

The tree of Figure 73 has been partially indexed so that classification of '4KCTG' and 'HPNWD' can be demonstrated. Classifying 'HPNWD' involves activating all indices that match a symbol of this string. N_{14} is indexed by three of these symbols, 'H', 'P', and 'N'. 'H' and 'P' are unique to N_{14} , predicting this node with probability, 1.0. 'N' predicts N_{14} with probability 0.67. The total predictiveness of 'HPNWD' is $1.0 + 1.0 + 0.67 = 2.67$. This exceeds the total predictiveness of the only other indexed node, the leaf containing 'HPNWD' which is not shown in Figure 73, but which is indexed by 'W' and 'D'. 'W' is unique to this string and therefore indexes it with probability 1.0, while 'D' indexes it with probability 0.5. The time required to reach N_{14} is $\frac{1}{2.67} = 0.37$. Since N_{14} only includes instances of category A, 'HPNWD' can be verified as a category A member having reached N_{14} .

Classification of '4KCTG' is somewhat more complicated than the previous example. Two symbols of this string, 'T' and 'G', are norms at the root and do not aid classification. '4' and 'C' predict N_1 with a combined score of 1.80. 'K' predicts N_6 with a score of 0.5. In this case, N_1 is initially selected to classify '4KCTG', where this node is reached in $\frac{1}{1.80} = 0.55$ time units. However, N_1 includes a mix of category A and B members; a correct prediction of category membership is not assured having only reached N_1 . There are several choices of what to do at this point. One is to continue classification from N_1 until a node is reached where a category can be guessed with assuredness. A second alternative is to simply generate a prediction of category membership and hope it is correct. Depending on the distribution of category A and B members this may lead to a significant number of erroneous predictions. As chapter 5 argued, taking advantage of norms may improve correctness, but classification must proceed to nodes with norms. As it turns out, no such category norms exist at N_1 , and so continuing classification becomes the choice of preference.

At N_1 , '4' and 'C' are normative and do not aid continued classification. 'K' predicts N_6 with probability 1.0, while 'T' and 'G' each indicate the leaf corresponding to '4KCTG' with probability 0.50 (not shown). Thus, N_6 and the leaf each have total predictiveness scores of 1.0. Each node is reached from N_1 in 1.0 time unit. At either node, '4KCTG' can be asserted as a member of category A. Both nodes are reached in the same time, but for convenience the following computations assume that N_6 is where '4KCTG' is classified as a member of category A.

The total time required to recognize '4KCTG' as a member of category A is the sum of the times required to reach N_1 and the time required to then reach N_6 . The total predictiveness for the first leg of this trip (i.e., Root to N_1) is 1.8, while the predictiveness of the second leg (i.e., N_1 to N_6) is 1.0. The time required

Class A instance	Inter-category similarity	Total predict score(s)	Total distance to prediction	Model time to prediction (ave.)	
4KCTG	high	1.8,1.0	2.0	1.55	1.52
GKNTJ		0.67	1.0	1.49	
4KC6D	medium	1.8,2.0	2.0	1.05	0.71
HPNSJ		2.67	1.0	0.37	
HPC6B	low	2.0	1.0	0.5	0.44
HPNWD		2.67	1.0	0.37	

Table 27

Human and computer model response times (3)

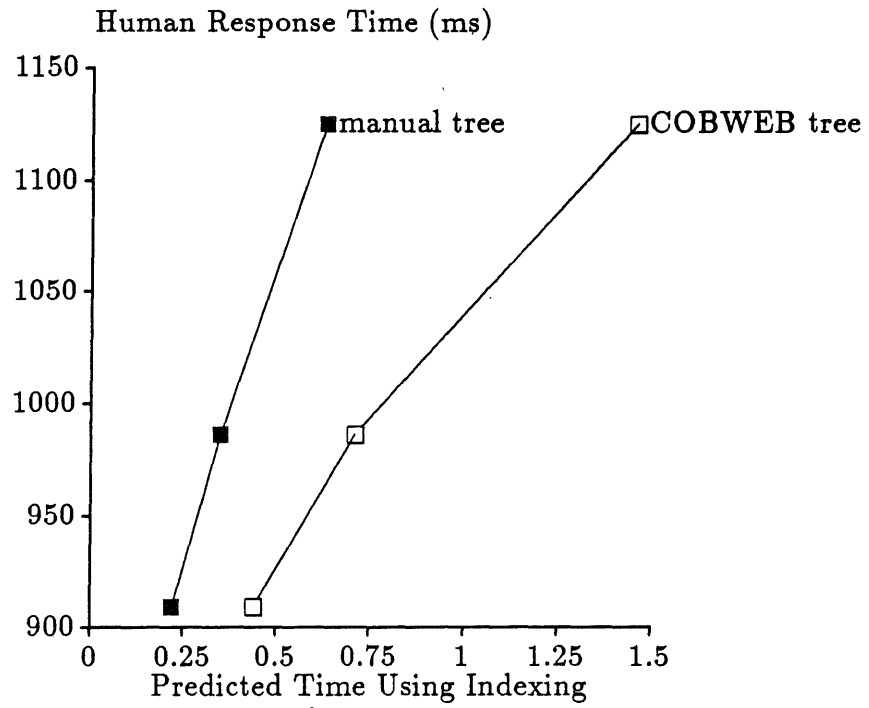


Figure 74

Human and computer simulated response times

for the first leg is $\frac{1}{1.8} = 0.55$ and the time for the second leg is $\frac{1}{1} = 1.0$. Thus, the total time required to reach N_6 and make a prediction of category membership is 1.55.

Table 27 shows the time required for each category A member to be recognized as such. In the case of two instances this requires traveling two nodes deep in the tree of Figure 73. Each of these instances are classified with respect to N_6 where they can be unambiguously identified as members of category A. In these cases, Table 27 shows the total predictiveness scores for each leg of the trip. The remaining nodes are first classified with respect to N_{14} where they can be identified as category A members. Figure 74 graphically shows that indexing the COBWEB classification tree, as well as the tree resulting from manually partitioning category A and B members, results in simulated classification times that are consistent with human performance.

7.4.1.3 Summary

Experimental studies indicate that human subjects verify category membership more quickly for some instances than for others. Rosch and Mervis [Ros75B] demonstrate that response time in a target recognition task depends on the degree that an instance is similar to members of the target category and dissimilar to members of a contrasting category.

This section demonstrates that indexing based on category utility yields simulated classification times that are consistent with human differences. Instances that have few values in common with other category members tend to have values that are unique within the category. Indices corresponding to values of such an object will be spread over a number of tree levels, thus diffusing the collective force with which any one node is indicated. This can slow the classification of the object with respect to all nodes. On the other hand, the attribute values of instances having much in common with contrasting category members will tend to be less predictive of target concept members; these values may even be normative, in which case they do not aid classification at all. It is also possible that target category objects may be grouped at tree nodes with members of a contrasting category.

Having reached such a node during classification, it may be necessary to proceed further down the tree, thus increasing response time, or be satisfied with making a guess at the node, thus increasing the probability of an erroneous prediction.

While a demonstration of the consistency of indexing with typicality effects is important, this discussion suggests a model-independent way of identifying typical and atypical instances. Indices are directed at nodes that maximize the value's collocation. Objects that share many values with other members of the category and have few values in common with members of contrasting categories will tend to be grouped together at classification tree nodes. Objects that most strongly indicate these nodes will have more values that index the node – more values for which collocation is maximized at the node. The next section demonstrates that a simple function of collocation is an accurate predictor of object typicality. It is an objective measure that is not tied to any specific model of recognition.⁵¹

7.4.2 Collocation as a Measure of Typicality

Rosch and Mervis [Ros75B] argue that the typicality of an object, with respect to a category, increases as the number of category members that share the object's properties increases. Inversely, the more properties the object shares with members of contrasting categories, the less it is typical of the target category. Rosch and Mervis hypothesize that typicality is predicted by a family resemblance measure which is a function of an instance, I , a category, C , and a set of contrasting categories, $\neg C$. Recalling function 3-1,

$$\text{family resemblance}(I, C, \neg C) = f\left(\sum_{C_i \in C} |I \cap C_i|, \sum_{C_j \in \neg C} |I \cap C_j|\right).$$

⁵¹ The distinction between a model-independent predictor of typical instances and a computational model of typicality is important and harkens back to the distinction between *specification* and *design* raised in chapter 1. Collocation is a specification of typicality, whereas indexing and recognition are computational processes that account for typicality.

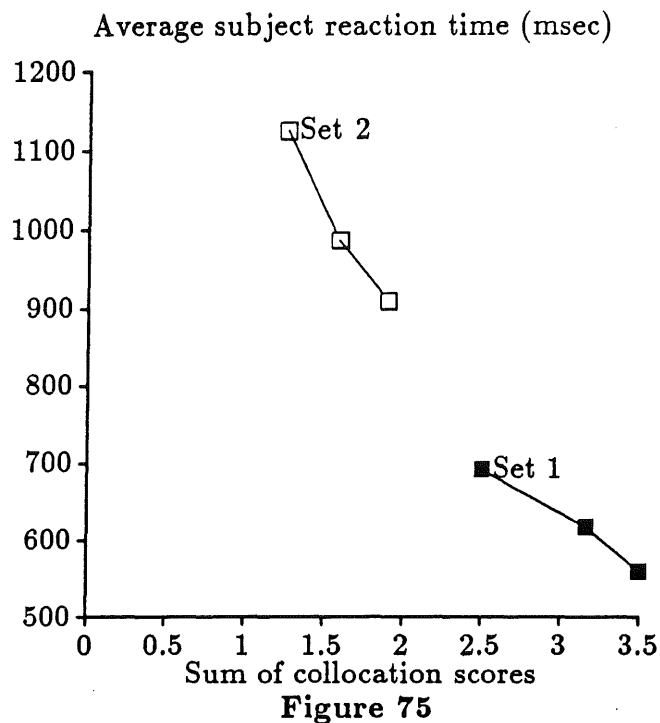
	Set 1			Set2		
	Letter String	Within-Category Overlap	Sum of collocation scores	Letter String	Between-Category Overlap	Sum of collocation scores
A	JXPHM	15	2.50	HPNWD	0	2.00
	QBLFS	15	2.50	HPC6B	1	1.80
	XPHMQ	19	3.17	HPNSJ	2	1.68
	MQBLF	19	3.17	4KC6D	3	1.51
	PHMQB	21	3.50	GKNTJ	4	1.34
	HMQBL	21	3.50	4KCTG	5	1.18
B	CTRVG			8SJKT		
	TRVGZ			8SJ3G		
	RVGZK			9UJCG		
	VGZKD			4UZC9		
	GZKDW			4UZRT		
	ZKDWN			MSZR5		

Table 28

Artificial categories used in typicality studies by Rosch and Mervis

While their studies indicate the importance of within- and between-category overlap, Rosch and Mervis do not propose a function that instantiates function 3-1. However, a candidate measure of typicality is the collocation measure proposed by Jones [JONE83]. While Jones proposed collocation as a predictor of the basic level, there are good reasons to believe that it qualifies as a good measure of typicality.

Recall that the collocation of an attribute value, $A_i = V_{ij}$, with respect to a category, C_k , is expressed as $P(A_i = V_{ij}|C_k)P(C_k|A_i = V_{ij})$. Note that as an attribute value's frequency within C_k increases, so does $P(A_i = V_{ij}|C_k)$ increase. Similarly, as the frequency of a value in contrasting categories increases, $P(C_k|A_i = V_{ij})$ decreases. For an object represented as $(A_1 = V_{1j_1}, A_2 = V_{2j_2}, \dots, A_m = V_{mj_m})$, the sum of object-value collocations with respect to a category, C_k , is



Recognition time as a function of collocation (typicality)

$$\sum_i P(A_i = V_{ij_i} | C_k) P(C_k | A_i = V_{ij_i}) \quad (7-1)$$

For reasons already mentioned, the sum of collocations instantiates the Rosch and Mervis notion of family resemblance. To verify the plausibility of function 7-1 as a measure of typicality, it is helpful to compare the measure with experimental evidence. For this purpose, the studies conducted by Rosch and Mervis [Ros75B] are most relevant.

As described, Rosch and Mervis used two sets of data to test the effect of varying intra- and inter- category overlap on typicality. These sets are given again in Table 28 as sets 1 and 2, respectively. After teaching subjects to distinguish categories A and B, Rosch and Mervis used a target recognition task to determine which strings subjects regarded as most typical (classified most quickly). The results of this task are graphed in Figure 75. The graph shows that as collocation

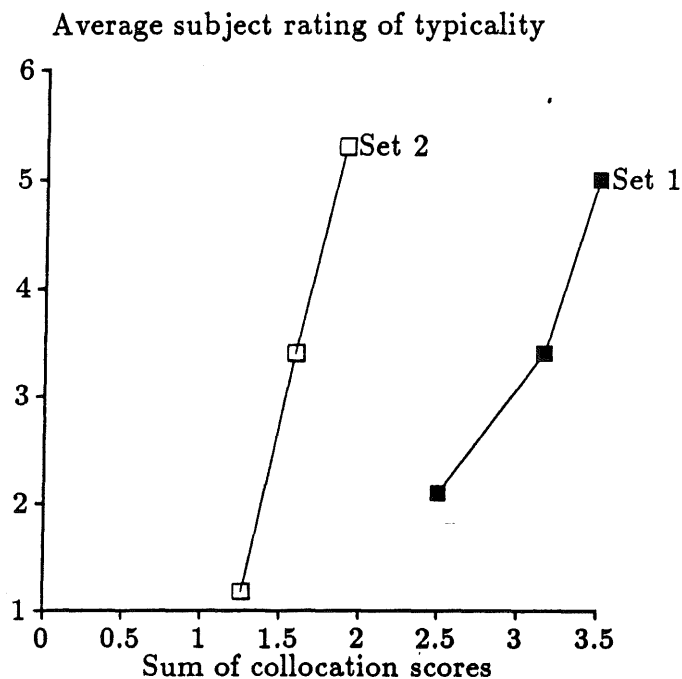


Figure 76

Subject's judgements of typicality and collocation

increases for a string, the time required to recognize the string as a member of category A decreases. Thus, collocation appears to be well correlated with recognition time.

As well as using the target recognition task to determine typicality, Rosch and Mervis also had subjects give a subjective judgement of typicality (using a 6 point scale). Figure 76 shows that subject judgements of typicality increase with increasing collocation within the same data sets. The lack of a (rough) linear relationship between collocation and subject judgements across data sets can perhaps be attributed to a conscious scaling or normalization on the part of subjects. This is in contrast to the continuum across data sets that can be seen in Figure 75, where no such normalization could occur. The target recognition and subjective judgement data bolsters claims that collocation reflects important determinants of typicality.

Category	Letter String	Within-Category Overlap	Between-Category Overlap	Sum of collocation scores
A	JXPHM	15	0	2.50
	XPHMQ	19	1	3.03
	PHMQB	21	2	3.14
	HMQBL	21	3	2.89
	MQBLF	19	4	2.33
	QBLFS	15	5	1.53
B'	GVRTC			
	VRTCS			
	RTCSF			
	TCSFL			
	CSFLB			
	SFLBQ			

Table 29

Within- and between- category overlap are simultaneously varied

One weakness of the previous analysis is that within- and between- category overlap are never in conflict in the sets shown. That is, either within- or between- category overlap is held constant, while the other is varied. In fact, Rosch and Mervis did conduct an experiment in which both dimensions were simultaneously varied. In particular, category B of set 2 was replaced by category B', and subjects were trained to distinguish categories A and B' shown in Table 29. No specific data regarding recognition times was published for this experiment, in large part because many of the differences were not statistically significant. This appears to be consistent with the the ranking of category A members using collocation, which illustrates that within- and between- category overlap act at cross-purposes.

Summing collocations for object values instantiates Rosch and Mervis' family resemblance function by rewarding within-category overlap and between-category non-overlap. Further, the collocation measure appears to rank objects in a manner similar to that obtained from human subjects with respect to various typicality

tasks. As a result, the collocation summation measure (7-1) will be used throughout this chapter as a measure of typicality.

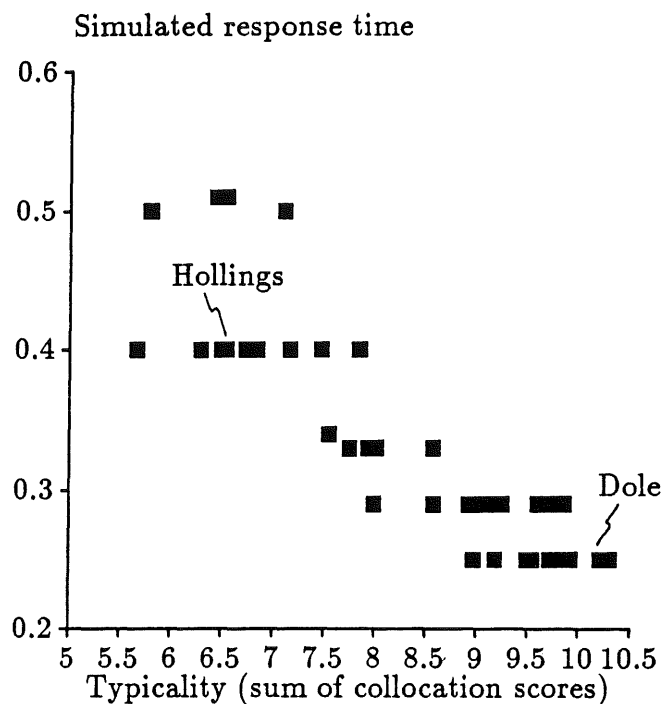
7.4.3 Typicality Effects in the Congressional Domain

With collocation as a measure of typicality it is possible to make predictions about human behavior in domains that were not previously used in psychological studies. In the remainder of the chapter, the sum of collocation function is used as a measure of typicality in several computer experiments. These experiments suggest possible human behaviors, some of which have been demonstrated in humans in *other* domains. However, with respect to the particular domains used for upcoming computer experiments, the results of computer simulation are speculative.

Consider the congressional domain discussed in chapter 5. One hundred senators were represented in terms of 14 key votes. COBWEB was run on this data without knowledge of the political parties (i.e., Democrat or Republican) of individual senators. COBWEB clustered senators into groups that corresponded roughly to 'liberals' and 'conservatives'.

7.4.3.1 Target Recognition

Indexing was imposed on the congressional classification tree produced by COBWEB. All members of the 'liberal' and 'conservative' clusters were reclassified using the indexed tree. The simulated time required to recognize each instance as a member of its respective cluster was recorded. Notice that this task required recognition with respect to a node of the tree (e.g., 'liberal') and not with respect to an externally defined class (e.g., democrat) as was the case using the artificial domains of Rosch and Mervis [Ros75B]. However, the constraints of the task are identical. In particular, the simulated time to recognize a particular senator as a 'liberal' is given by the quotient of the 'liberal' node's distance from the root (i.e.,



Response times for 'conservative' senators as function of typicality

1) and the total predictiveness over all indices that match a value of the senator being classified.

Figure 77 shows the simulated time required to recognize senators classified under the 'conservative' node as a function of their collocation (i.e., typicality) with respect to the class of conservative senators. Similarly, Figure 78 shows the simulated time required to recognize members of the 'liberal' cluster. Each point on these scatter graphs corresponds to a single senator. Each graph indicates a strong tendency for more typical instances of each class to be recognized more quickly. Because each attribute (i.e., vote) has only two possible values, 'yes' or 'no', few values are relatively unique to subsets of the 'liberal' and 'conservative' classes. For this reason, most of the response time variances are *not* due to a diffusion of predictiveness across tree levels, but result from values being shared with members

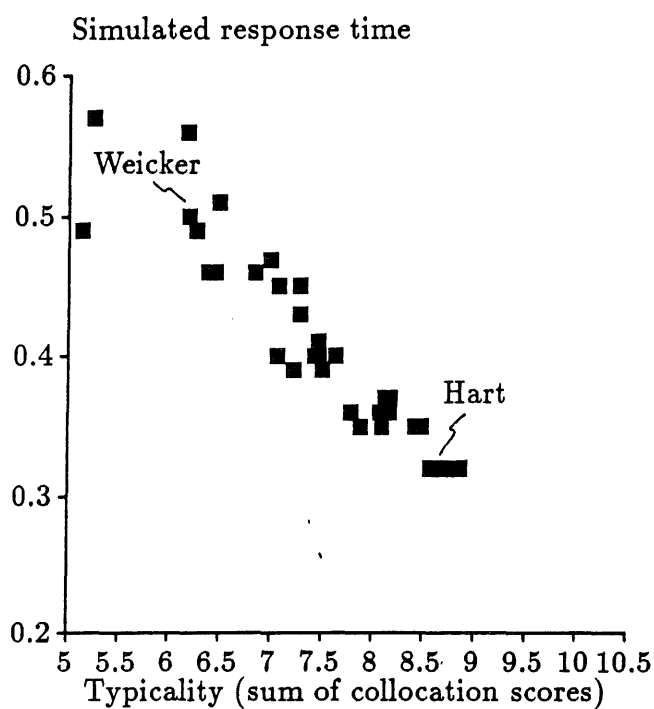


Figure 78

Response times for 'liberal' senators as function of typicality

of the contrast category. That is, atypical 'liberals' vote like 'conservatives' along some dimensions.

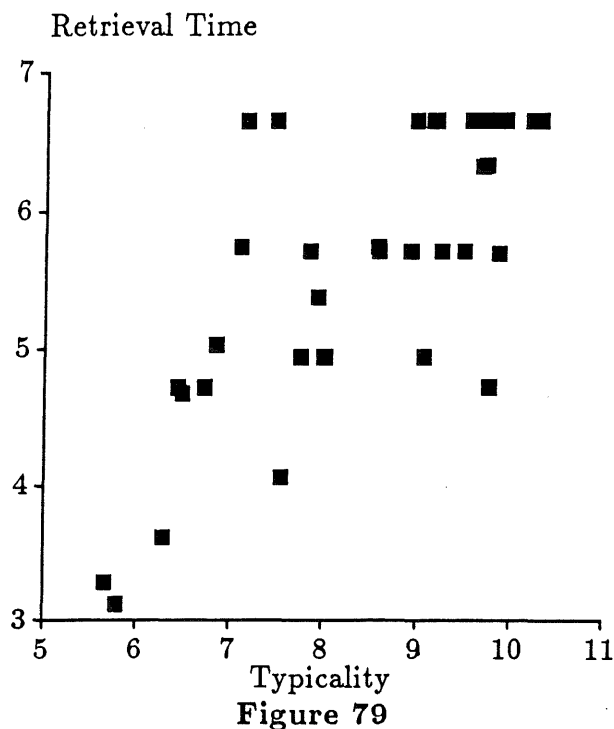
As earlier computer experiments with the Rosch and Mervis data indicated, demonstrations of typicality effects are not limited to categories that correspond to classification tree nodes. For example, individual Democrats and Republicans also vary in terms of their typicality with respect to these externally defined categories. In the classification tree produced by COBWEB, Democrats that share many values with other Democrats and few values with Republicans will tend to be placed under the same nodes. A typical Democrat will more strongly predict a node with a concentration of Democrats (i.e., the 'liberal' node) than an atypical Democrat. Atypical Democrats may even predict nodes with a majority of Republicans (i.e., the 'conservative' node). Similar principles guide the grouping and recognition of typical and atypical Republicans.

Recognition of atypical Democrats may require going to deeper levels of the tree if a correct prediction is to be assured; this results in slower recognition times. An alternate strategy is to guess political party when it becomes normative. This will lead to more errors in predicting political party for atypical Democrats. For example, an atypical Democrat may be regarded as a Republican if it is initially classified at the 'conservative' node. Most likely, recognition of atypical objects with respect to externally defined categories involves some tradeoff between deep classification, resulting in slower times, and using normative values to cut search off, resulting in erroneous predictions for atypical instances.

7.4.3.2 Exhaustive Retrieval

Several overviews of typicality data [SMIT81, ROSC78, MERV81] contend that when asked to list all members of a category, human subjects will list instances roughly in order of typicality. The common wisdom seems to be based on two studies. Rosch [ROS75A] provides some evidence that superordinate concept (e.g., *furniture*) members are listed in order of typicality. Rosch, Simpson, and Miller [ROS76B] taught subjects classes of nonsense strings and found that after learning, instances tended to be listed in order of typicality. Computer experiments using the indexing scheme strongly disagree with results from human studies. The purpose of this section is to explore the impacts of this apparent contradiction.

A number of strategies for exhaustive retrieval using this chapter's model were tried in an effort to match psychological results. In general, each retrieval strategy generalized the recognition procedure used in previous experiments. Previous computer experiments assume that object recognition proceeds by following indices corresponding to object values and summing index weights, $P(N_k | A_i = V_{ij})$. A procedure to deal with partial object descriptions (not used in previous experiments) followed indices for all values of missing attributes as well. However, for missing attributes, the probability of a particular value's truth, $P(A_i = V_{ij})$, was



Time to retrieve 'conservatives' in the congressional domain

used to weight the predictiveness of the value, $P(N_k|A_i = V_{ij})$. Rather than terminating at a single leaf, all leaves that are consistent with the partial object description are returned. If all attributes are missing, this is recognition of the null object. All leaves under a specified node are consistent with the null description. 'Recognition' of the null object is one model of exhaustive retrieval. The time of retrieval is based on summing $P(A_i = V_{ij})P(N_k|A_i = V_{ij})$ for all indices emanating from the specified node.

Figure 79 shows that when retrieving members of the 'conservative' node of the congressional tree, if there is any tendency, it is that atypical 'conservatives' are retrieved first. No retrieval strategy that was reasonably close to the above led to agreement with apparent human behavior. In the best cases there was no correlation between typicality and retrieval order. Intuitively, these results stem from an important property of hierarchical representation in general and this

chapter's particularly; normative values are not used to index lower-level nodes. The most important attribute values in determining object typicality are factored out of the retrieval process. A consequence of using a hierarchical representation is that more typical objects will not necessarily be retrieved faster, unless additional assumptions are made.

Disagreement with psychological studies indicates that either the model's explanation of exhaustive retrieval is incorrect, or that psychological findings must be qualified. Rosch, Simpson, and Miller [Ros76B] found that when subjects learned (by examples) classes of nonsense strings, instances tended to be retrieved in order of typicality. However, subjects often retrieved nonsense strings that were never taught. This observation suggests the possibility that instances were not explicitly stored, but only a summary representation of each class was remembered. In this case, instances have to be *generated*, rather than retrieved from the summary description. Generation of previously unseen instances could be expected. Additionally, if the summary representation is similar to the sort of representation at nodes in a COBWEB tree, generation would be influenced by attribute value distributions. It is probable that objects would be generated beginning with the most typical objects. Findings of this experimental study may not generalize to cases in which there is reason to believe that instances (or subclasses) are explicitly stored.

A second study [Ros75A] indicated that members of two superordinate classes are listed roughly from most to least typical. However, this finding may not generalize to all classes. In general, superordinate classes share few properties across subclasses and many times are defined in terms of function, rather than perceptual properties. While computer experiments certainly cannot negate these findings, they suggest that they be more thoroughly investigated. In cases where class instances (or subclasses) are explicitly stored, the hierarchical representation

of this chapter offers intuitively good reasons to believe that atypical objects may be retrieved first. Informal experiments with two UCI computer science graduate students seem to bear this out. Each was asked to list all the birds they knew; ostriches, hawks, eagles, and penguins were among the initially retrieved instances. This is only intended as anecdotal support for the predictions of the computer model. It should be recognized that more extensive human experimentation is required to identify weaknesses with the computer model and/or the common wisdom stemming from psychological experiments.

7.5 Basic Level and Typicality Effects

The previous sections have looked at how indexing explains basic level and simple typicality effects. This section looks at interactions between these classes of phenomena. In particular, sufficiently atypical objects may be initially recognized with respect to nodes that are subordinate to the basic level. Also, typicality effects may not emerge during target recognition when the target concept is subordinate. Apparently, this latter phenomenon cannot be verified (or disconfirmed) by current data on human behavior. However, it is a prediction of the model.

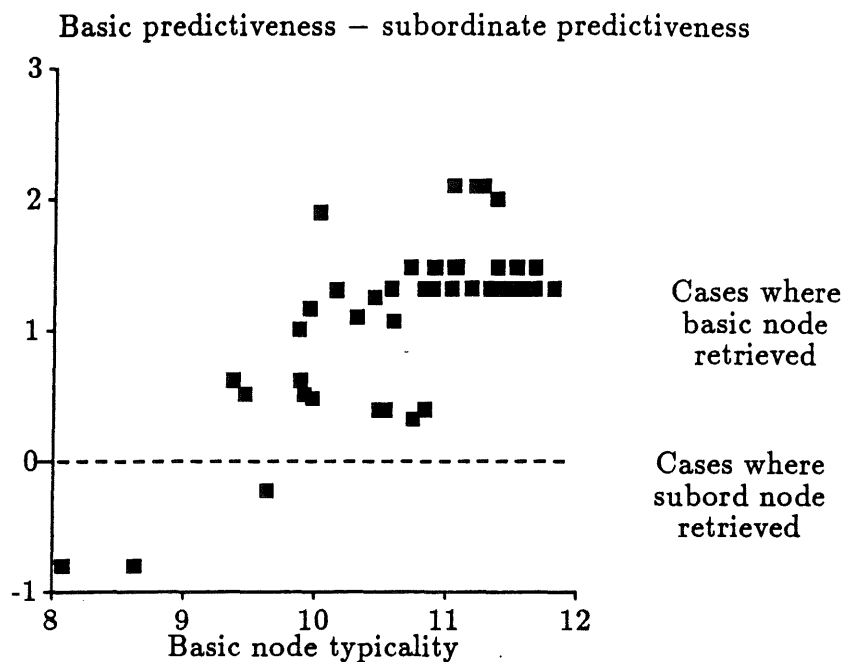
7.5.1 The Impact of Typicality on Basic Level Effects

Psychological experiments indicate the tendency of human subjects to initially classify observations with respect to the basic level of a hierarchical classification scheme. Demonstrations in the early part of this chapter indicate that a properly indexed classification tree offers one explanation for this behavior. However, experiments by Jolicour, Gluck, and Kosslyn [JOL184] qualify the human preference for basic level concepts. In particular, they found that atypical instances of a basic level class were sometimes initially recognized with respect to a subordinate concept. For example, while a robin (i.e., typical *bird*) is usually recognized as a *bird*,

a penguin (i.e., atypical *bird*) is recognized as a *penguin*. This section illustrates how and why the indexed memory model accounts for this phenomenon.

Experiments with Rosch's nonsense strings indicated that an object's typicality can effect recognition time. In the case where an object contains relatively unique values with respect to other target class members, recognition may be slowed because indices are spread across several levels of a tree; in other words, indices may be spread across the tree vertically. Atypicality may also imply that competing concepts cause indices to be diffused horizontally. Both factors reduce the total predictiveness towards the target node and presumably this slows recognition time. The effects of horizontal and vertical dispersion of indices can also effect recognition time with respect to the basic level. For example, objects that are not typical of their basic level class may be more similar to competing classes than typical objects. This will cause the basic node to receive less total predictiveness when classifying an atypical object. Atypical objects of a basic class may also have relatively unique values compared to other class members. This will result in a simultaneous decrease in the total predictiveness towards the basic node and an increase in the total predictiveness of subordinate nodes. If the total predictiveness of an atypical object towards the basic level is sufficiently weakened, while being sufficiently increased towards a subordinate node, the object may be initially recognized with respect to the subordinate node.

The impact of typicality on basic level effects is demonstrated using the tree constructed by COBWEB in the thyroid domain. Each member of node 'N1' of this tree was classified using the tree after adding indices. The total predictiveness towards the 'N1' (i.e., basic) node was recorded for each thyroid case history. In addition, the total predictiveness of the most predicted subordinate node was recorded. Each point on the scatter graph represents the difference between the total predictiveness of the basic node and the most predicted subordinate. A



negative score on the vertical dimension indicates that an object displayed a greater predictiveness towards a subordinate node than to the basic node. In this case, three members of 'N1' have a greater total predictiveness towards a subordinate node, and are thus recognized by the lower-level node first.

While atypical objects may be recognized with respect to subordinate nodes, in most cases their total predictiveness towards the basic node is not sufficiently weak and total predictiveness towards subordinate nodes is not sufficiently strong to disrupt the basic level preference. In most domains, there are no exceptions to the basic level preference, even for the most atypical objects. Figure 81 shows that while the difference tends to less for atypical objects of the 'liberal' node, total predictiveness is always greater for the 'liberal' node. Members of the 'liberal' class are always recognized first as liberals (i.e., the basic node).

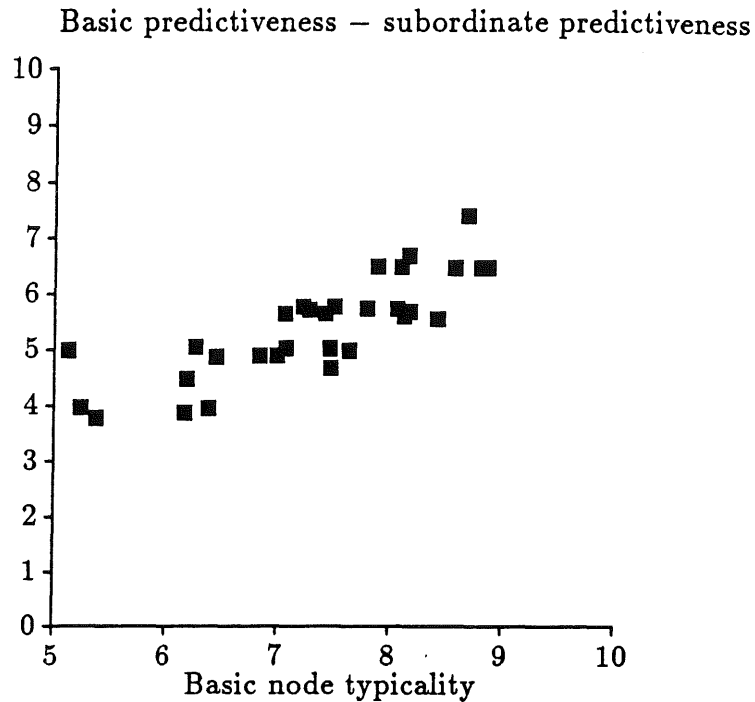


Figure 81

Difference between basic and subord total predictiveness (liberal)

7.5.2 The Effect of Target Concept Generality on Typicality

Apparently there have been few studies of the impact of target concept generality (i.e., superordinate, basic, subordinate) on typicality effects. Most psychological studies of typicality in natural domains make the (stated or unstated) assumption that the target concept resides at the basic level. For example, Rosch and Mervis [Ros75B] run target recognition experiments using concepts that are verified to reside at the basic level (e.g., *car*, *chair*, *lamp*).

Target recognition using basic concepts limits the apparent impact of other concepts on recognition, since these basic concepts lie at the entry points of a concept hierarchy. An implication of hierarchical classification is that recognition with respect to subordinate concepts can be significantly influenced by higher level organization, since an object must first be recognized with respect to the basic level and perhaps other concepts before reaching a subordinate target concept.

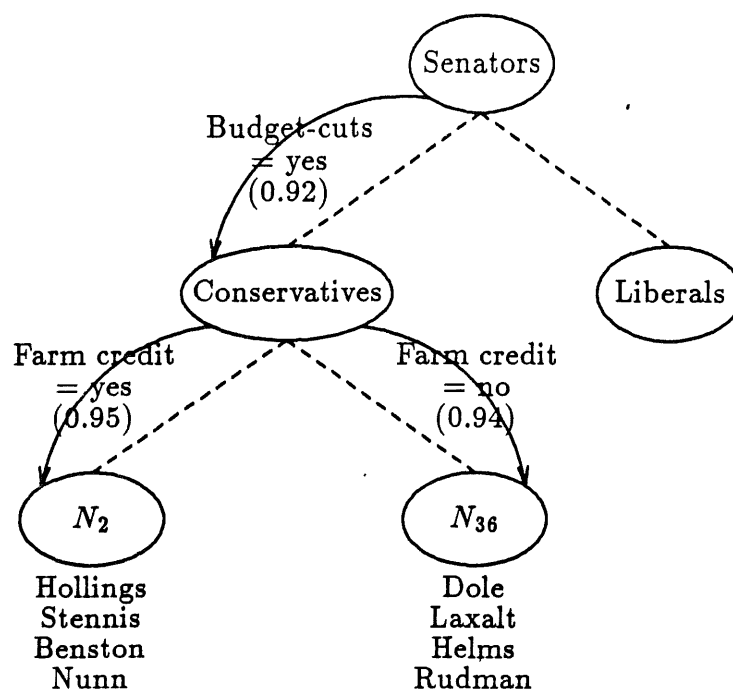


Figure 82

The 'conservative' class with subordinate target concepts

As an example of subordinate target recognition, reconsider part of the congressional classification tree shown in Figure 82. Included are the 'conservative' node and its two children, N_2 and N_{36} . Listed under each subordinate are a number of senators classified under that node. These nodes differ considerably in structure. N_2 contains all ten Democrats classified under the 'conservative' node. Additionally, the average collocation of a member of N_2 with respect to the 'conservative' class is 7.75. In contrast, the average collocation of N_{36} members with respect to 'conservatives' is 9.48. Intuitively, N_2 tends to have atypical 'conservatives', while N_{36} classifies more typical 'conservatives'. Properties of these subordinates impact the object recognition process, particularly influencing typicality effects associated with each class.

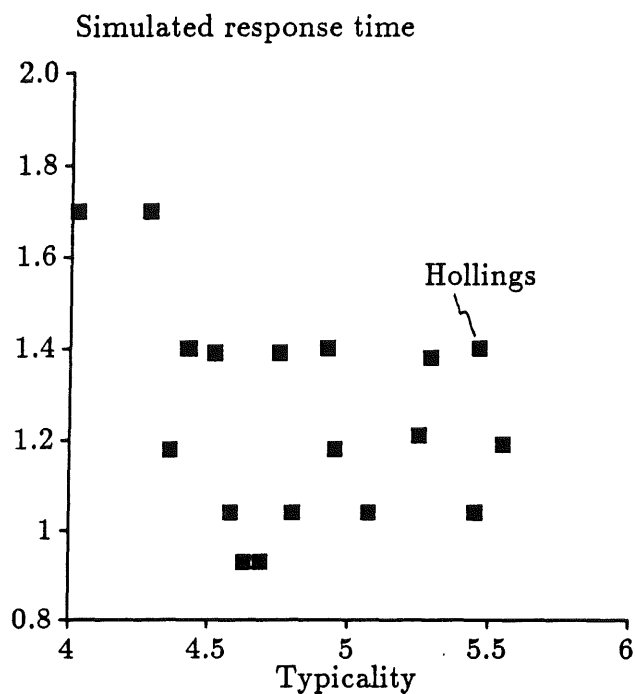
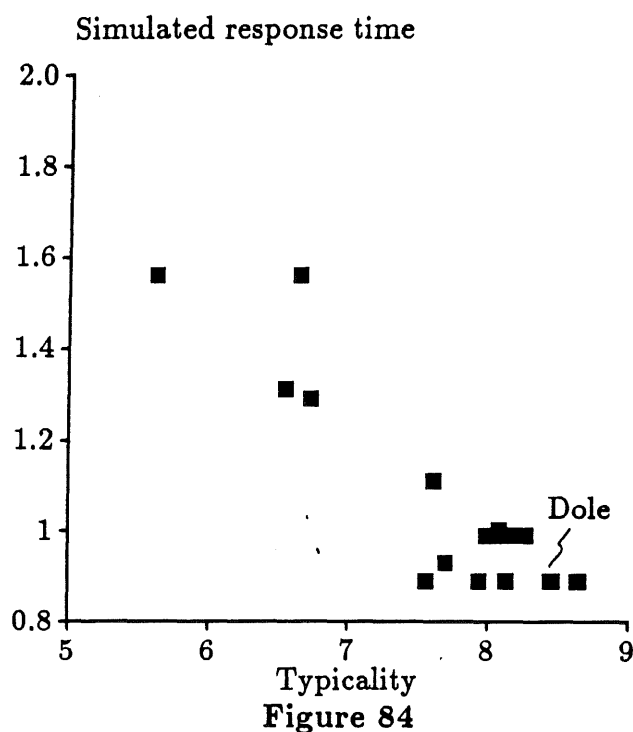


Figure 83

Response time as function of typicality to subordinate node (N_2)

Figure 83 shows the simulated time required to recognize each member of N_2 . Each object was first recognized with respect to the top level node, 'conservatives'. Thus, distance in the time calculation was 2 for each object. For example, 'Hollings' was recognized as a member of N_2 in time, 1.40. This is relatively slow, despite the fact that 'Hollings' is a relatively typical member of N_2 . In general, Figure 83 does not reveal any response time trends as a function of subordinate class typicality. Apparently this results because a typical member of N_2 will tend to be an atypical 'conservative'. Since recognition passes through the 'conservative' node, atypicality with respect to 'conservatives' may offset any response time gain because of typicality with respect to the subordinate class.

In contrast to members of N_2 , Figure 84 shows that N_{36} instances follow the expected trend; response time decreases with an increase in typicality with N_{36} . Unlike N_2 , N_{36} instances tend to be typical members of the 'conservative'



Response time as function of typicality to subordinate node (N_{36})

class. In general, atypicality with respect to the basic node does not counterbalance typicality with respect to the subordinate.

The explanation of typicality offered in this chapter focuses on object recognition in the context of a larger conceptual organization. This leads to interference effects between levels of generality, notably basic and subordinate levels. Recognition of typical subordinate concept members is not necessarily faster if these objects are atypical members of higher level concepts. However, this section has not investigated typicality with respect to *superordinate* concepts. Rosch and Mervis [Ros75B] report a study of human superordinate typicality effects, but they looked at subjective judgements of typicality and not response time. A hypothesis suggested by the memory model of this chapter is that interference between basic level and superordinate concepts are minimal. Although objects would be first recognized at the basic level, a superordinate concept is reached by climbing an

IS-A link and not by another evidence combination process. Thus, response time differences would be dominated by an object's typicality with respect to its basic level concept.

7.5.3 Discussion

Two interactions between basic level and typicality effects have been hypothesized and modeled. First, the basic level preference may not emerge for sufficiently atypical objects. Second, an object's basic level typicality may offset typicality with respect to subordinates. Considering object recognition in the context of an organization of concepts, rather than considering concepts in isolation, makes these interactions apparent.

This chapter assumes that basic level and typicality effects stem from the same principles of memory organization and recognition. Object typicality is postulated to vary with an object's predictability and predictiveness with respect to a class. In particular, the sum of $P(A_i = V_{ij}|C_k)P(C_k|A_i = V_{ij})$, or *collocation*, for all values of an object predicts typicality with respect to a class, C_k . Typical objects have higher scores by this measure and are more forcefully directed to the appropriate nodes. Typical instances can also be viewed as the best representatives of a class. Similarly, basic level nodes are those that tend to maximize collocation for the most frequently occurring attribute values tend to be most forcefully predicted nodes over all objects of the environment. Basic level classes are 'generalized instances' that best represent the class of all domain objects.

7.6 Caveats

Indexing based on category utility explains a number of important effects, but its scope as a cognitive model must be qualified. The model's account of typicality and basic level effects depends on 'direct' observations of an object's attribute values, e.g., through visual input. However, there has been much work done using

verbal cues. Recognition models that assume verbal input make an assumption that concepts are initially retrieved by an unspecified process. For example, the question may be asked, "Is a *robin* a *bird*?" While this chapter has made no effort to model experimental findings using verbal cues, it is useful to discuss their possible impact on this chapter's memory model.

Several studies of target recognition use verbal cues for both instances (or sub-concepts) to be recognized and target concepts. Once retrieved via verbal cue, recognition with respect to the target can proceed by comparing attribute values of the two concept definitions, as in the general summing recognition procedure of chapter 3. This avoids the problem of how concepts interact in a larger memory structure, but it may be an accurate model of how two concepts are compared once they are extracted by a verbal cue. A second alternative is to climb IS-A links from a subclass (e.g., *robin*) to the target (e.g., *bird*) [COLL69, SMIT81]. This model predicts that subclasses will be more quickly recognized with respect to more immediate superordinates. Recent evidence [SMIT81] indicates that while this is generally true, there are exceptions. These findings pose a number of questions about the hierarchical representation scheme of this chapter. Because object properties are assumed to be directly perceivable, this chapter has modeled target recognition as a top-down process. Unanswered, however, has been whether the representation accounts for psychological effects (particularly typicality) using the bottom-up processing implied by reliance on verbal cues.

In the animal domain, most types of birds will be recognized more quickly as a *bird* than as an *animal*. However, studies indicate that the name of an atypical bird (e.g., *chicken*) may be more quickly recognized as an animal than as a bird. If climbing IS-A links is the only way of accounting for these effects, then this chapter's current classification tree organization offers no explanation of why certain atypical subclasses are more quickly recognized as more distant superordinates.

Currently, there is one path from any node up through its ancestors; immediate superordinates must be encountered first. Smith and Medin [SMIT81] suggest allowing subclasses to have multiple IS-A links. Thus, links may go from *chicken* to *bird* and from *chicken* to *animal*. A still more general structure would allow overlapping concepts; a *chicken* could be a *bird*, as well as a *farm-animal*. Smith and Medin argue that IS-A links may be of varying 'lengths' and traversal time is proportional to 'length'. Presumably, 'length' is dependent on the extent to which the classes share features, but this is largely unspecified and offers few constraints for augmenting the cognitive model.

Explanations for the above data need not be dependent on the existence of IS-A links. In fact, proposals for weighting IS-A links [SMIT81, COLL75, COLL69] may assume that weights reflect the underlying similarity between concepts; IS-A links may be viewed as a convenient approximation of an upward evidence combination procedure, perhaps similar to the downward-directed one presented in this chapter. However, such a procedure will not be specified here.

In summary, the indexing scheme of this chapter accounts for typicality effects with the caveat that target recognition occurs with objects that are explicitly represented by their properties; no explanation is given for the case where subclasses are indicated symbolically by words or otherwise.

7.7 Chapter Summary

This chapter describes an indexing scheme for classification schemes that accounts for certain basic level and typicality phenomena. Apparently, this is the first computational model of basic level effects. The model also predicts certain interactions between basic level and typicality effects. Inherent in the model's explanation of these phenomena are a number of claims that vary in generality. In

general, the model's commitment to these claims decreases with their generality. Specifically, these claims are given in order of generality and commitment.

- Basic level effects can be best explained in the context of hierarchical concept organizations.
- Less obviously, typicality effects must emerge from object recognition in organizations of concepts. Typicality effects cannot be adequately explained by a simple comparison of concept definitions outside the context of a larger memory.
- Interactions between basic level and typicality effects can be expected. In particular, an object's typicality with respect to its basic level class can impact whether it is first recognized at the basic level. Second, typicality effects with respect subordinate nodes are dependent on an object's typicality with respect to more general classes.
- Recognition time is inversely related to the degree that an object's values predict a class. Predictiveness has been formalized in terms of cue validity.
- Recognition is constrained to following attribute-value indices. Indices are directed only to nodes that maximize collocation, a tradeoff of attribute value predictiveness and predictability.

This chapter has ignored the problem of memory update. While the indexing model presented herein has been compared with human experimental results, a weaker, but important test of its plausibility as a description of human memory is its ability to support learning. Chapter 8 examines the problem of memory update using the indexing scheme of the chapter. Computer experiments in the next chapter suggest that the constraint that indices be directed only at collocation maximizing nodes is too restrictive. In the context of learning, this constraint leads to a 'brittle' memory structure. A general 'fix' to the problem is sketched and instantiated.

Chapter Acknowledgements

Conversations with Kurt Eiselt regarding output order during exhaustive retrieval were also quite helpful.

CHAPTER 8

COBWEB/2: Incremental Update of Indexed Memory

8.1 Chapter Overview

Chapter 7 showed how indexing can be imposed on a COBWEB classification tree. In addition to simulating psychological phenomena, intuition and computer experimentation indicates that indexing tends to classify objects in the same manner as an explicit category utility calculation. This chapter extends the scope of this discussion to include the feasibility of using and maintaining indices during memory update.

This chapter describes a derivative of COBWEB, COBWEB/2, that incrementally builds indexed classification trees. COBWEB/2's behavior and output are characterized along many of the same dimensions as COBWEB, including the utility of trees for inference, update cost, the ability to find optimal classification trees, and convergence time. The findings of this chapter indicate that COBWEB/2 is generally an effective learner. Prediction accuracy using indexed classification hierarchies constructed by the system approximate the levels achieved by COBWEB. An important qualification to these findings is that COBWEB/2 is overly brittle in the early stages of tree construction.

Because effective inference under dynamic memory conditions is a high level characteristic of much of human memory, an investigation of indexed memory's abilities in this regard necessarily impacts claims of the model's psychological consistency. Many *speculative* [HALL85] research efforts would take COBWEB/2's general abilities at learning and prediction as positive evidence for the legitimacy of indexed memory as a cognitive model. However, this chapter carefully avoids such

claims. Rather, all confirming evidence for the legitimacy of indexed memory was given by the *empirical* treatment of chapter 7. Instead, the view of this chapter is that speculative analyses primarily provide disconfirming evidence, if they provide any evidence at all – if a memory structure cannot be used for learning, this casts doubt on its feasibility as a tool for human, as well as purely artificial intelligence. In particular, the finding that COBWEB/2 is brittle early in the learning process is taken as evidence against the strictly enforced indexing scheme of chapter 7. However, a simple fix to COBWEB/2 diminishes learning brittleness as a stumbling block to claims of the indexing scheme's psychological consistency.

In summary, the primary goal of this chapter is to qualify claims of the psychological consistency of the indexing scheme of chapter 7. The objective of this chapter is not to present a polished concept formation system. In general, the analysis of COBWEB/2 reflects a view that speculative investigations are primarily useful for supplying disconfirming, not confirming, evidence for psychological plausibility.

8.2 The COBWEB/2 Algorithm

The basic COBWEB/2 algorithm and data structures closely parallel those of COBWEB. Chapter 7 described the form of COBWEB/2 classification trees. Probabilities of the form $P(A_i = V_{ij} | N_k)$ are stored with each V_{ij} at tree nodes, N_k . These probabilities are measures of attribute value predictability and collectively represent probabilistic concepts. Classification tree nodes in COBWEB/2 are identical in form to those produced by COBWEB. Additionally however, probabilities of the form $P(N_k | A_i = V_{ij})$ are stored with values that index nodes. These probabilities are measures of value predictiveness. Indices are only directed at nodes that maximize the collocation of the indexing value. In addition to value

indices, COBWEB/2 assumes IS-A links connect a node to its parent and CHILD links connect a node to its children.

COBWEB/2's heuristic measures for guiding classification tree construction have also been described previously. Function 7-3 (i.e., $\sum_i P(A_i = V_{ij})P(N_k|A_i = V_{ij})P(A_i = V_{ij}|N_k)$) is used to determine which existing node, N_k , best hosts a new object. Recall that this function approximates the classification choices indicated by category utility (4-4) for existing classes. However, because merging and new class creation involve the introduction of new classes that impacts partition size, these operators use the complete category utility function (4-4). The remainder of this section focuses on the operators used by COBWEB/2 to incorporate objects. This operator set closely mirrors COBWEB's.

8.2.1 Placing an Object in an Existing Class

The procedure for placing an object in an existing class is exactly that given in chapter 7. That is, indexing initially activates a number of possibly relevant nodes and the total predictiveness of each activated node is computed. The predictiveness score of a node, N_k , is given by $\sum_i P(N_k|A_i = V_{ij_i})$, where V_{ij_i} is a value of the object being classified that also indexes N_k . Of the activated nodes, the node, N_{max} , with the greatest predictiveness score is kept for more thorough evaluation. In addition, all nodes that intersect in at least one indexing value with N_{max} are kept for evaluation. In many cases there are no nodes that intersect with an indexing value of N_{max} , i.e., N_{max} is the only node kept.

Nodes that are kept for evaluation need not be immediate children of the root, but may be nodes at some intermediate level of the tree. These nodes are guaranteed to be nonancestral. The best-host from among this set is the one that maximizes $\sum_i P(A_i = V_{ij_i})P(N_k|A_i = V_{ij_i})P(A_i = V_{ij_i}|N_k)$. While chapter 7 assumed that best-host is almost always N_{max} , this second evaluation stage always occurs in COBWEB/2.

```

FUNCTION COBWEB/2 (Object, Root ( of classification tree ))
IF Root is a leaf
THEN Return expanded (and indexed) leaf to
    accommodate new Object
ELSE
    • Index and evaluate indexed nodes and identify Best-host.
    • COBWEB/2 (Object, Best-host)
    • UPDATE-COUNTS(Object, Best-host)
    • UPDATE-INDICES(Object, Best-host)
    • UPDATE-INTERMEDIATE-NODES
      (Object, Parent-of(Best-host), Root)
    • UPDATE-COUNTS(Object, Root)
    • UPDATE-INDICES(Object, Root)

```

```

PROCEDURE UPDATE-INTERMEDIATE-NODES
    (Object, Descendent, Ancestor)
IF Descendent ≠ Ancestor
THEN UPDATE-COUNTS(Object, Descendent)
    UPDATE-INDICES(Object, Descendent)

```

Table 30

COBWEB/2 Pseudocode that assumes classification is only operator

8.2.1 Updating Intermediate Nodes

If the only operator for update is classification with respect to existing classes, classification recursively proceeds until a leaf is reached. In general though, classification may jump tree levels. For example, upon first entering the tree, Best-host need not be the child of the root. Because of this, nodes that reside between Best-host and the root must be updated. This involves updating the counts and indices of intermediate nodes.

Pseudocode for applying the classify operator is given in Table 30. Note that immediately after identifying Best-host, COBWEB/2 is recursively called to classify the object with respect to lower levels. Counts and indices of Best-host are not updated until after an object has been classified with respect to lower levels. Updating Best-host's counts simply increments appropriate attribute value counts to reflect the addition of the new object. After updating Best-host's

counts and those of lower level nodes (on the recursive call to COBWEB/2), UPDATE-INDICES updates the downward pointing indices from Best-host for all values of the incorporated object. In general, this may involve computing the collocation for each object value for all descendants of Best-host (including Best-host itself). If all subnodes need to be investigated, this can be a relatively expensive operation, particularly if it needs to be done for every node encountered during object incorporation. Statistics presented later suggest that this is not necessary when updating the hierarchy an object at a time.

Once Best-host has been identified, nodes lying between Best-host and Root are updated from the bottom up using UPDATE-INTERMEDIATE-NODES. This procedure updates the counts of each intermediate node to reflect the addition of a new object. In addition, the indices that emanate from each intermediate node are updated as well. Finally, the counts and indices of Root are updated. In COBWEB/2, node count update occurs after lower-level nodes have been updated. This is in contrast to COBWEB, which updates counts as classification descends the hierarchy. The more complicated procedure used by COBWEB/2 is required if counts are to be consistently maintained, even when classification jumps levels.

8.2.2 Creating a New Class

In COBWEB/2, creation of new classes is controlled by evaluating the impact of the new class on a partition. In particular, a new class is created as a child of the root if it results in a better partition (by category utility) than adding the object to the *best existing child*. This is exactly the same rule that is used in COBWEB, but in this case, the best host may not be a child of the root. The best existing child of the root is found by climbing IS-A links from the best host, as shown in Figure 85. In many cases, the best *host* identified by indices will also be a child of the root, i.e., best *host* and best *child* will be the same node. The quality of the

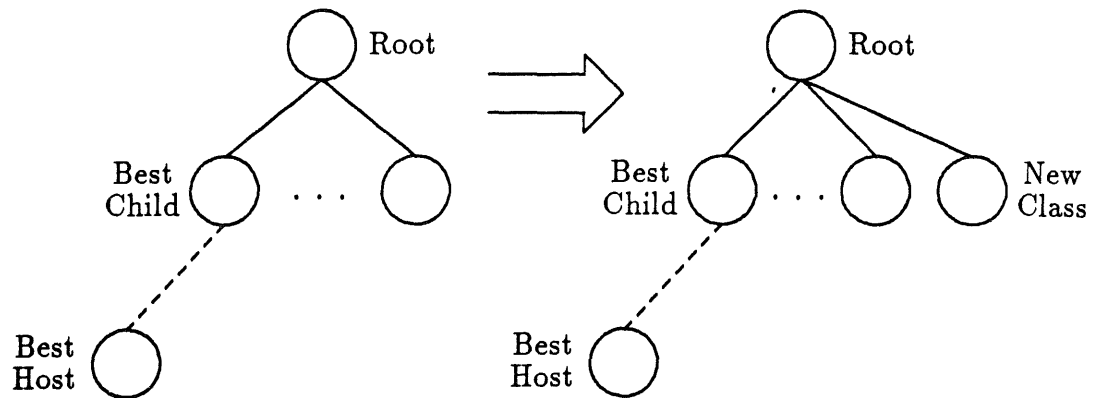


Figure 85

Testing the quality of creating a new class in COBWEB/2

IF NEW-CLASS-CONDITION (Object, Best-host, Root)
 THEN Make Object a child of Root. Indices are
 directed to the new class in the normal
 course of updating Root indices.

FUNCTION NEW-CLASS-CONDITION (Object, Best-host, Root)

- 1) Find the ancestor of Best-host that is a child of Root.
 This ancestor is called Best-child and may be Best-host.
 The other children of Root are called Other-children.
- 2) IF $CU(\{\text{Best-child}\} \cup \text{Other-children} \cup \{\text{Object}\})$
 $\geq CU(\{\text{Best-child} + \text{Object}\} \cup \text{Other-children})$
 AND $\{\text{Object}\}$ maximizes collocation of at least one value
 THEN RETURN TRUE
 ELSE RETURN FALSE

Table 31

Consider creating a class and do so if appropriate

partition formed by adding the object to the best existing *child* is compared with the quality of the partition formed by creating a new class.

Finally, since indices are being used, it is important that a newly created node maximize collocation for at least one value. This will insure that it is indexed and can be accessed in the future. The complete rule for creating a new node is given in Table 31.

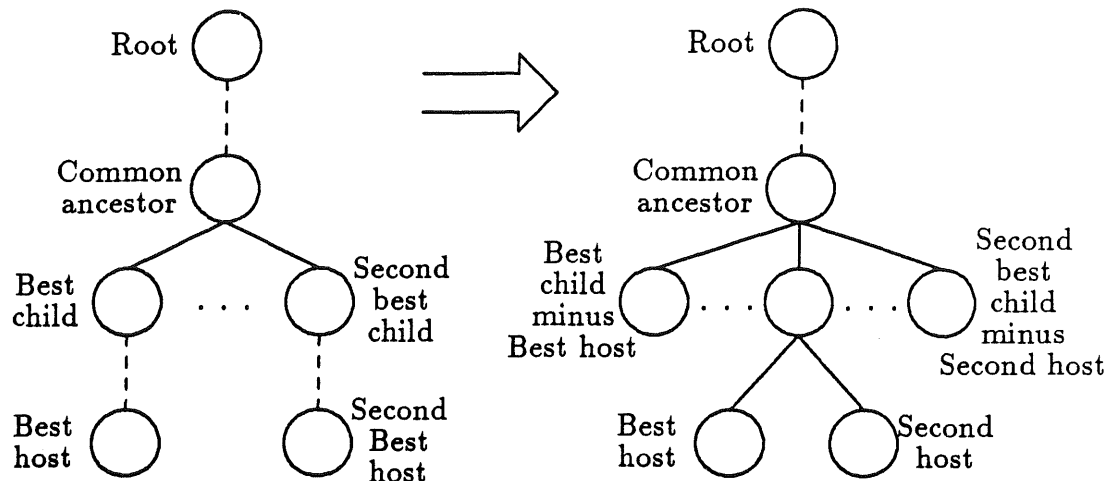


Figure 86

Merging the two best hosts in COBWEB/2

8.2.3 Merging and Splitting

Like COBWEB, merging and splitting are allowed in COBWEB/2. However, unlike COBWEB, the two best hosts may not be children of the root and may even reside at different tree levels. In COBWEB/2 merging and splitting are regarded as a single operator; merging the two best hosts may require first splitting apart nodes that were previously at different parts of the tree, at least as defined by IS-A and child links. Merging nodes in COBWEB/2 is similar, but not identical, in form to node merging in COBWEB *with superordinate nodes*.

Evaluating indexed nodes, N_k , by 7-3 orders the nodes by the degree that they match an incoming object. The node with the highest score is the best existing host. In addition, the second highest scoring node (if one exists) is the second best host. COBWEB/2 considers merging the two best hosts.

Figure 86 demonstrates how merging of the two best hosts is evaluated and performed. The most specific common ancestor of the two hosts is identified by climbing IS-A links and the *best* and *second best children* of this common ancestor are identified. If best host is a child of the common ancestor, *best host* and *best*

IF MERGE-CONDITION (Object, Best-host, Second-best-host)
 THEN 1) Remove Best-host and Second-best-host from their respective
 ancestors up to their Common-ancestor
 2) Merge Best-host and Second-best-host and make the Merged-node
 a child of Common-ancestor. Add indices to Merged-node.
 3) Recompute indices for all former ancestors of Best-host and
 Second-best-host beginning with their former parents until
 reaching Common-ancestor.

FUNCTION MERGE-CONDITION (Object, Best-host, Second-best-host)
 1) Find the Common-ancestor of Best-host and Second-best-host.
 Identify the children of Common-ancestor, Best-child and
 Second-best-child, that are ancestors of Best-host and
 Second-best-host, respectively. The other children of
 Common-ancestor are called Other-children.
 2) IF $CU(\{\text{MERGE-NODES}(\text{Best-host, Second-best-host})\}$
 $\cup \{\text{Best-child} - \text{Best-host}\}$
 $\cup \{\text{Second-best-child} - \text{Second-best-host}\}$
 $\cup \text{Other-children}\}$
 $\geq CU(\{\text{Best-child} + \text{Object}\}$
 $\cup \{\text{Second-best-child}\}$
 $\cup \text{Other-children}\}$
 THEN RETURN TRUE
 ELSE RETURN FALSE

Table 32

Consider merging best hosts and do so if appropriate

child will be the same node. A similar observation is true of *second best host* and *second best child*. If either one of these conditions is true, *best* and *second best child* are merged, without regard to how this effects partition quality. The philosophy is that if a similarity is noticed and there is no 'competing' similarity, there is no harm in merging the similar classes together. In this manner, *superordinate* classes can be formed in a much more natural way than described in chapter 4.

In cases where both *best host* and *second best host* are not immediate children of *common ancestor*, a simple merging of these nodes may not be appropriate. The reason is that *best* and *second best host* already participate in different superordinate categories that may have been formed because of similarities different from the one

```

FUNCTION COBWEB/2 (Object, Root ( of classification tree ))
IF Root is a leaf
THEN Return expanded (and indexed) leaf to
accomadate new Object
ELSE Determine best and second best existing hosts
and perform one of the following:
  a) Consider creating a new class and do so if appropriate.
  b) Consider merging the two best hosts and do so if
  appropriate.
  c) IF none of the above (a or b) were performed
      THEN COBWEB/2(Object, Best host)
           UPDATE-COUNTS(Object, Best-host)
           UPDATE-INDICES(Object, Best-host)
           UPDATE-INTERMEDIATE-NODES
             (Object, Parent-of(Best-host), Root)
           UPDATE-COUNTS(Object, Root)
           UPDATE-INDICES(Object, Root)

```

Table 33

Second approximation of operator control in COBWEB/2

that currently indexes them. The best hosts can still be merged, but only if merging them (and removing them from intermediate ancestors) results in a better set of children of the common ancestor. Figure 86 illustrates this procedure. Merging is evaluated using the complete category utility measure (4-4) and is applied to an object set partition (i.e., the children of the common ancestor). The rule used to merge nodes is given in Table 32.

Pseudocode for controlling the three operators of classification, creation, and merging are given in Table 33. This pseudocode adds class creation and merging to the earlier control structure, which only assumed classification. The abbreviated mention of new class creation and merging expand into the more precise rules described earlier for each operator.

8.2.6 Promoting and Dropping Subtrees

Like COBWEB, COBWEB/2 regards some nodes as *useless*. A *last-accessed* count is maintained for each node. If the node does not classify an object after a variable threshold number of objects, a prototype of the node is used to determine if the node is useless for purposes of future classification.

In COBWEB/2, *unreachable* nodes may also occur. Unlike COBWEB trees, classification trees produced by COBWEB/2 use attribute-value indices that are directed at nodes that maximize collocation. When a node ceases to maximize collocation for a value because of changes in the tree, the index is moved to the appropriate (collocation maximizing) node. Associated with each node is a POINTED-AT field. The POINTED-AT field contains the number of indices that currently point at the node. Whenever a new attribute-value index is directed at a node, the node's POINTED-AT value is incremented. This field is decremented whenever an index is directed away from the node. Unreachability occurs when a node's POINTED-AT count reaches zero; this means that no indices point to the node and it cannot be activated during indexing.

When a node's POINTED-AT count reaches zero, it is deleted – a process similar to *garbage collection* [STAN80]. All attribute value indices that emanate from the deleted node are removed and the pointer counts of subordinate nodes are decremented as appropriate. This is analogous to promoting the children of the deleted node. However, decrementing subordinate node pointer counts may cause some of these to reach zero as well. Very often, once a node's pointer count reaches zero, a chain reaction causes all subordinate node pointer counts to reach zero as well. In this case, all descendants of the deleted node are deleted as well. This procedure is similar to that employed in UNIMEM [LEBO82], but this latter system does not allow indices to skip levels; thus it can automatically delete entire

subtrees without explicitly decrementing the POINTED-AT fields of lower-level nodes.

As with any node, when a leaf's pointer count reaches zero, it is deleted from the classification tree. However, deleting a leaf removes an actual object from the tree. In COBWEB/2, when a leaf is deleted, counts in superordinate nodes up to the root are decremented to reflect the object's removal.

8.2.7 Summary

The control structure for COBWEB/2 is close in form to COBWEB's. The major complicating factors are that indexing allows classification to skip levels. Top-down classification must be augmented with a process of updating nodes that were initially skipped. In many ways COBWEB/2 is similar to the version of COBWEB that built superordinate nodes. However, two things distinguish these systems. First, COBWEB/2 does not use different evaluation functions to modify superordinate and basic level nodes. Rather, COBWEB/2 introduces a superordinate node when any similarity is noticed through indexing. Second, COBWEB/2 does not *force* classification to originate at the basic level. Instead, it relies on the findings of chapter 7 that indexing naturally (and almost always) results in classification at the basic level. This latter characteristic, along with findings that COBWEB/2's heuristics typically identify the same best hosts as COBWEB's, lead to an initial hypothesis that the two systems have similar behavioral properties and result in similar classification schemes. Characterizations of COBWEB/2 that follow generally verify this intuition, but this analysis also uncovers some important qualifications.

The following sections characterize COBWEB/2 along the dimensions of update cost, classification tree quality, convergence time, and prediction accuracy. However, while these are the same dimensions used to analyze COBWEB, this chapter considers them in a different order. This reordering makes certain problems

with COBWEB/2 apparent at the beginning of analysis, thus illuminating more subtle matters later on.

8.3 COBWEB/2 as an Incremental Learner

Like its predecessor, COBWEB/2 is an incremental concept formation system that can be evaluated in terms of update cost, the quality of concept hierarchies, and the time required for a 'stable' classification to be achieved. This section evaluates COBWEB/2 along each of these dimensions, making comparisons with COBWEB as appropriate. Comparisons indicate advantages and problems with using indices for memory update and motivate two extensions to the basic algorithm.

8.3.1 Cost of Assimilating a Single Object

COBWEB/2 employs a two-step classification procedure. Update cost can be computed by looking at the overall costs of each stage of classification. That is,

$$\text{cost} = (\text{activation costs}) + (\text{evaluation costs}) \quad (8-1)$$

Initially, only classification with respect to an existing class is considered.

Activation costs can be approximated by

$$(\text{number of nodes activated}) \times (\text{single node activation cost})$$

Node activation is triggered by object recognition. Each object has one value along each attribute. If A is the number of attributes, the cost of activating a single node and computing its total predictiveness is $O(A)$. It is difficult to analytically come by the number of nodes that are activated. In domains used for experimentation however, the number of nodes activated at each level of classification seems to approximate the branching factor of the tree. For example, the average number of

nodes activated at the top level of the congressional tree is 2.6, while there are 2 nodes at the top level. Assuming that the branching factor, B , is a good average approximation of the number of activated nodes at each level and that classification proceeds to a depth of $\log_B n$ on average, the total number of activated nodes is about $O(B \log_B n)$. The total activation cost is

$$\text{activation costs} = O(B \log_B n \times A).$$

Evaluation costs can be approximated in a manner very similar to activation costs. The number of nodes that are evaluated is always less than or equal to the number of activated nodes. For example, the average number of activated nodes at the top level of the congressional tree was 2.6, while the number of nodes that needed to be kept for further evaluation was 1 in all cases. While evaluation in this second stage is more extensive than computing total predictiveness, it still is bounded by the number of defining attributes. That is, $P(A_i = V_{ij})P(N_k | A_i = V_{ij})P(A_i = V_{ij} | N_k)$ is computed for each object value, of which there is one per attribute. Accumulated activation costs and evaluation costs are given by the same upper bound approximation. Since their costs are additive, the total cost of update assuming only classification with respect to existing nodes is

$$\text{cost} = O(B \log_B n \times A).$$

Adding merging and class creation to the set of permissible operators does not appear to raise the upper bound approximation significantly. Class creation requires evaluating an object set partition using category utility at a cost of $O(BAD)$, where D is the average number of values per attribute domain. Merging inflicts the same cost since it examines a partition. After adding merging and new class creation, the update cost of COBWEB/2 is approximated by $O(B \log_B n \times AD)$.

This analysis has not considered the costs that can be incurred when a node's POINTED-AT count reaches zero. This state requires that all downward pointing indices from this node be removed, possibly causing a chain reaction in which the entire subtree is dropped. However, this generally occurs for very small subtrees early in processing and does not appear to affect the average update costs of COBWEB/2.

In general, COBWEB/2's update cost appears to average $O(B \log_B n)$, where B is the average branching factor of the tree and n is the number of previously classified objects. This undercuts COBWEB's cost of $O(B^2 \log_B n)$ somewhat. On average, it appears that indexing can effectively be used to cut down on the number of nodes that need be considered for object incorporation. However, unmentioned has been the cost of updating indices to point at collocation maximizing nodes. In the worst case, the addition of an object may require checking every descendent of a node. In the worst case, if the updated node classifies n objects, this may require checking $2n + 1$ nodes, which is the number of nodes in a complete binary tree.

Properties of index update were investigated experimentally in the domain of congressional voting records used in chapter 5. An experiment was designed to collect statistics on index update during learning. Shifts of indices that emanated from the root over the course of clustering senators were recorded. For example, an index for 'Budget-cuts = yes' might shift from one subordinate of the root to another to reflect a change in where collocation for that value was maximized after incorporating the voting record of senator Gary Hart. The variable of interest in this experiment was the degree of shift. If indices tended to point at the same nodes or shift locally, from a node to a child or parent, then this is an indication that a full blown search for collocation maximizing nodes is not required; only a local search in the immediate vicinity of the existing collocation maximizing node need be made following object incorporation. On the other hand, if collocation maximizing nodes

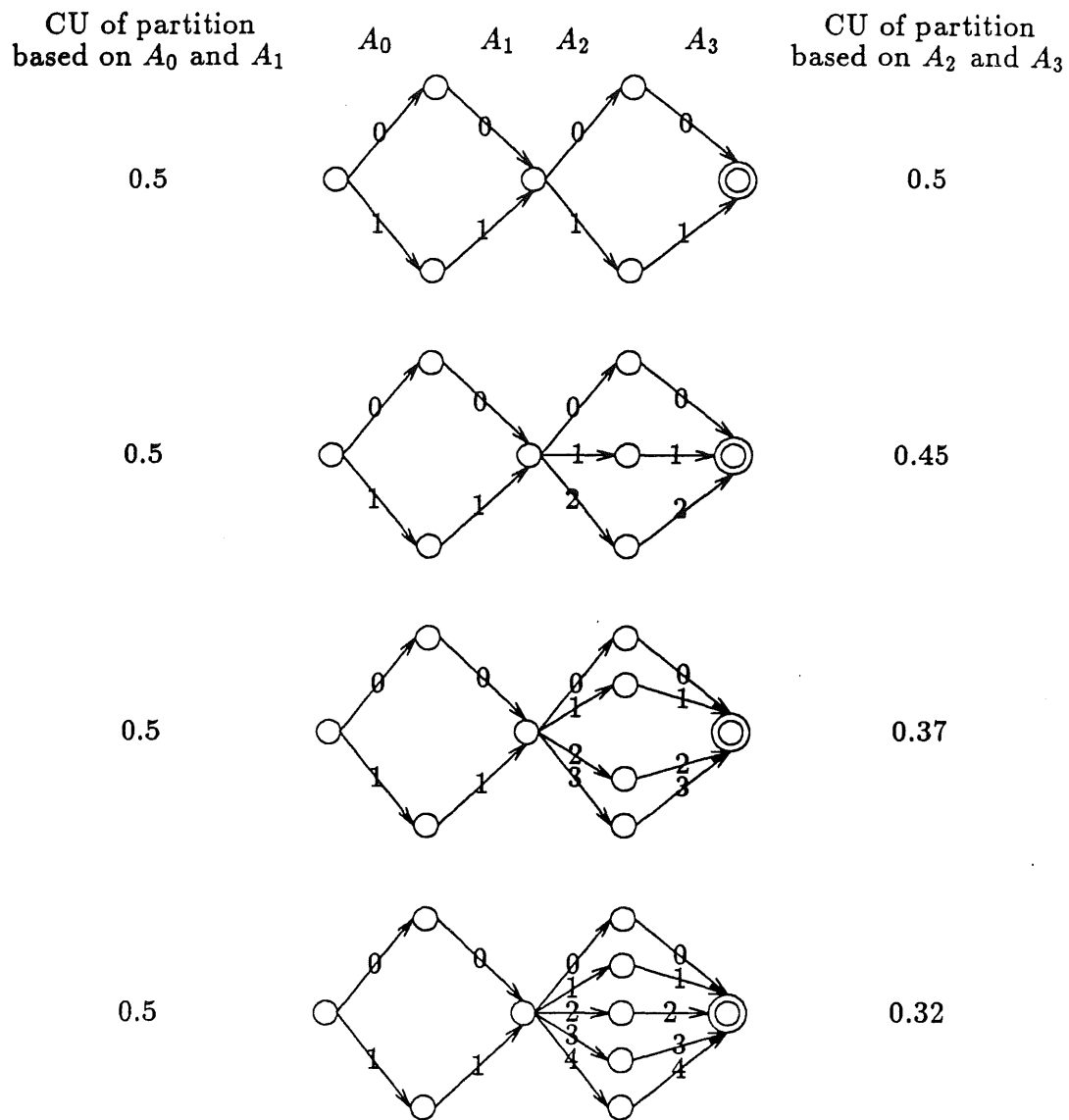


Figure 87

Domains with global and local optimums

are not localized, this implies that a more costly search is required following each update.

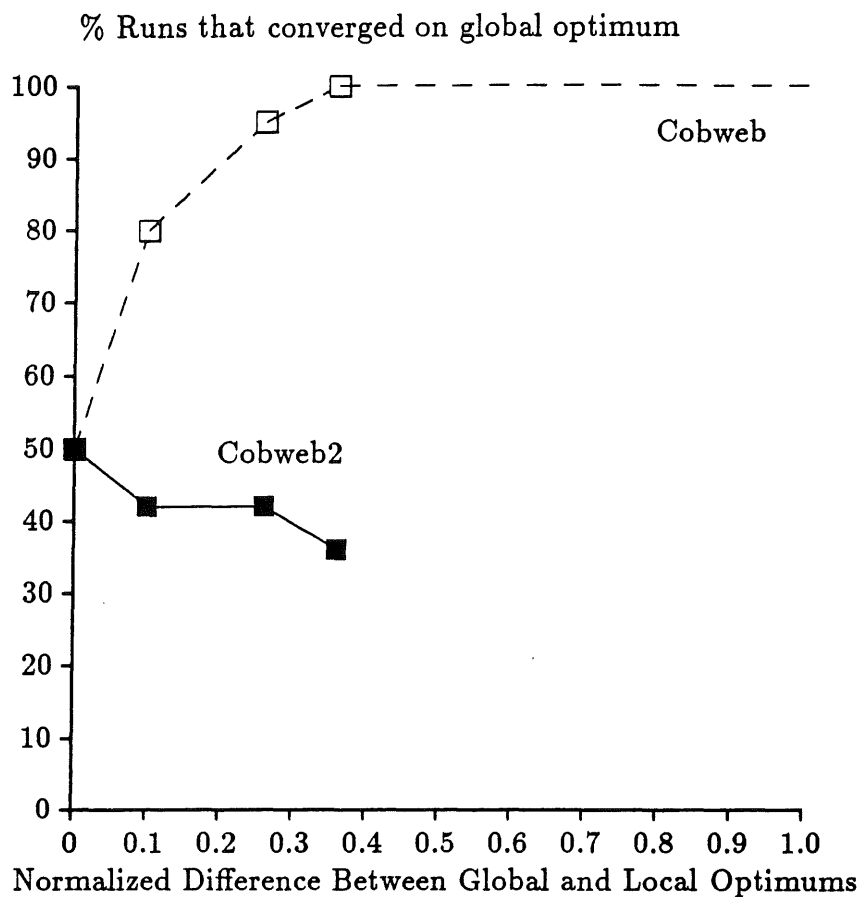
COBWEB/2 was run on the one hundred senator descriptions. Statistics were recorded for 110 updates (node unreachability/uselessness caused 10 senators to be reprocessed). In 58 of these cases there was no change in collocation maximizing

descendents with respect to any value. In the remaining 52 cases some shift in the node that maximized collocation for a value occurred. These cases can be further divided up. In 29 cases, a shift of one node occurred. That is, collocation maximization for a value shifted from a node to its parent or one of its children. This class of shifts is important because these constitute 'local' shifts; spotting them requires only a search around the immediate vicinity of the node that previously maximized collocation. However, in 29 instances (not mutually-exclusive from the last 29), a shift of greater than one node occurred for *one* attribute value. This class of shifts can not be spotted by a localized search (of 1 node).

Apparently, in 29 cases a local search around a previous collocation maximizing node could not guarantee that the new collocation maximizing node would be found. However, 15 of these cases occurred in the first quarter of the instances; in these early trials many values were being seen for the first time and changes through merging and new class creation were changing the structure of the first tree level. In contrast, only 3 of these shifts occurred in the last quarter of the instances, and none occurred in the last 18. A tentative lesson is that while extensive searches for collocation maximizing nodes may be required early in clustering, a search appears unnecessary after some stabilization has occurred. Despite the relative infrequency of nonlocal shifts in collocation maximizing nodes, the data above points to the possible costliness of maintaining indices early in the clustering process.

8.3.2 The Quality of Classification Trees

COBWEB/2's matching function (7-3) tends to place objects in approximately the same classes as using the complete category utility function. In fact, when the indexing scheme and classification procedure were developed in chapter 7 and imposed on trees constructed by COBWEB, object recognition proceeded correctly for each object. However, recall from section 7.2 that the ability of this scheme to approximate the behavior of COBWEB is somewhat dependent on the



Convergence on optimal partitions in COBWEB/2

number of previously observed objects. Moreover, apart from the use of different evaluation function, there may be problems that are introduced by using a discrete indexing scheme rather than a continuous evaluation function. This section more fully investigates the ability of COBWEB/2 to converge on trees with top levels that best partition the observed objects according to category utility.

As with COBWEB, the current system's ability to converge on optimal partitions was tested in the four domains of Figure 87. To review, each domain contained a globally optimal partition defined along attributes, A_0 and A_1 . A segregation based on the values of A_2 and A_3 define a partition of lesser quality, a locally optimal partition.

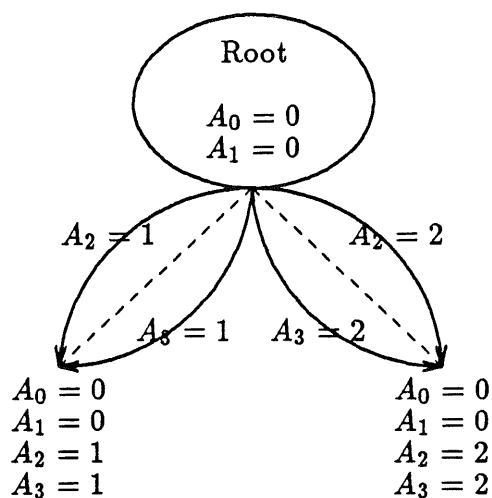


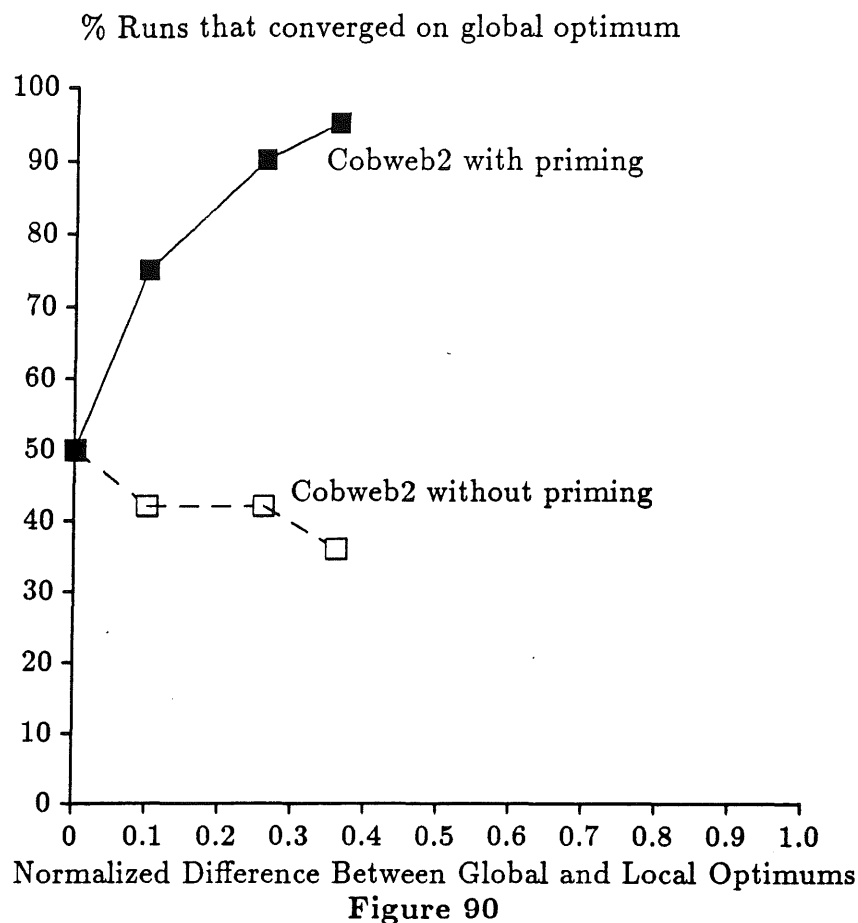
Figure 89

An example of indexing 'brittleness'

Figure 88 indicates that COBWEB/2 does not reliably converge on globally optimal partitions. In fact, as disparity between globally and locally optimal partitions increases, the ability of COBWEB/2 to converge on the global optimal decreases.⁵² This behavior is the opposite of that displayed by COBWEB, which rapidly tends to converge on globally optimal partitions as they become more obviously distinguished from other patterns in the data.

Intuitively, COBWEB/2's behavior is a result of a restrictive indexing policy. Consider the example of an initial clustering in Figure 89 over two objects in domain 3. These objects share values of 0 along attributes A_0 and A_1 . However, they are sufficiently distinct along A_2 and A_3 that they are initially placed in separate classes. However, $A_0 = 0$ and $A_1 = 0$ are not predictive of either node and these values are not used as indices, while the predictiveness of A_2 and A_3 values are used as indices. Already there is a bias against forming classes based

⁵² The difference between the category utility scores of the globally optimal partition and the partition of lesser quality were normalized to lie between 0 and 1, inclusive. This was done by dividing the difference of the absolute category utility scores by the absolute score of the globally optimal partition.



Convergence on optimal partitions using priming in COBWEB/2

on the values of A_0 and A_1 , though it is still possible that such a classification will arise. However, in general, the greater variability in the values of attributes A_2 and A_3 will cause them to be unique across the small set of initially created classes; they will likely serve as indices. In contrast, the less variability of A_0 and A_1 values will tend to cause objects with the same values along these attributes to be spread across initial classes. In many cases, before there is an opportunity to merge these classes together, A_0 and A_1 indices will be dropped.

COBWEB/2 can be easily thrown off track in the early stages of clustering. This appears to occur in domains where a number of competing patterns exist. Distinct, but infrequently occurring patterns will tend to spread out instances

exhibiting the dominant pattern early on. Indices that reflect the dominant pattern may be lost. The requirement that indices only be directed at collocation maximizing nodes is overly restrictive. Any 'fix' should relax the requirements for indexing early in the clustering process. At one extreme, all nodes could be indexed initially, giving dominant patterns a chance to arise. Perhaps the most straightforward approach in this regard is to use a continuous evaluation function early in processing. In particular, the COBWEB incorporation process is used initially. Indexing is utilized after a 'sufficient' number of objects are observed. In COBWEB/2 this number was rather arbitrarily specified to be 10. Not surprisingly, as Figure 90 indicates, allowing memory to be *primed* in this fashion results in behavior that is nearly identical to COBWEB's.

Priming is certainly not the last word in adapting indexing for problems encountered early in concept formation. It is unlikely, that human learners so abruptly move from a continuous to a more discrete indexing procedure. However, in human learning a gradual shift from continuous to discrete indexing is possible. Priming is a straightforward and inexpensive means of approximating this shift. COBWEB's continuous evaluation procedure is a rough functional equivalent of a more extensive indexing scheme and it allows memory to better find high quality partitions. Moreover, it is probable that the more conservative policies for partition update used by COBWEB will lead to more 'stable' trees. This may be important in reducing the extent of index shifting that plagued COBWEB/2 in the congressional domain. In the sections that follow, results from COBWEB/2 without priming are stressed, since these represent worst-case scenarios. With priming, COBWEB/2 can be expected to mimic COBWEB's behavior.

8.3.3 Number of Objects Required for Convergence

COBWEB/2 has trouble converging on optimal partitions when competing patterns are present in the data, but when there is one predominant pattern,

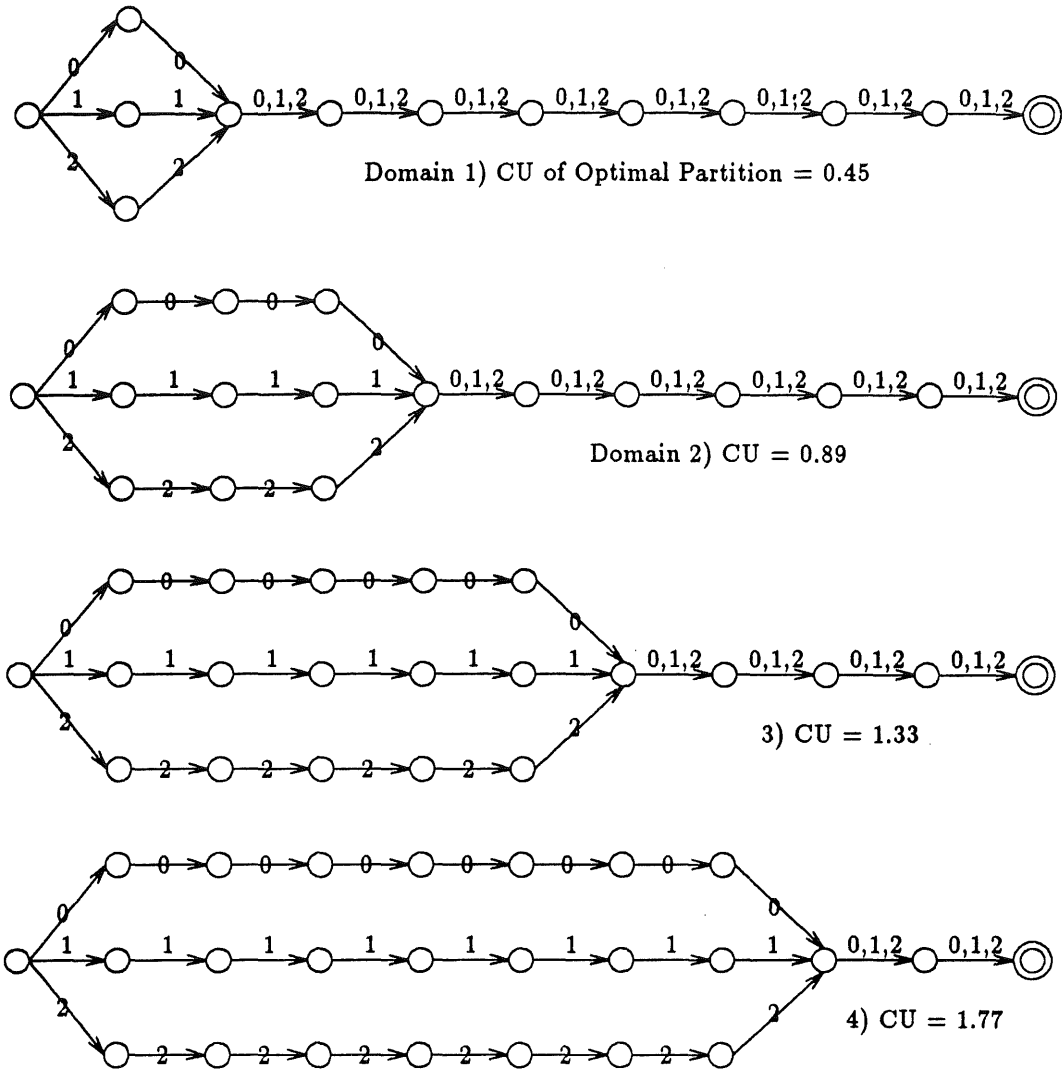


Figure 91

Domains used to test convergence time

the following experiments indicate that COBWEB/2 has little problem finding it. COBWEB/2 was tested in the four domains previously used in chapter 6, and given again in Figure 91. COBWEB/2 was run on five random orderings from each of the four domains. During learning, 100 objects were intermittently and randomly selected from the domain being learned and classified (but not incorporated) with respect to the classification formed thus far. If the top-most level of COBWEB/2's

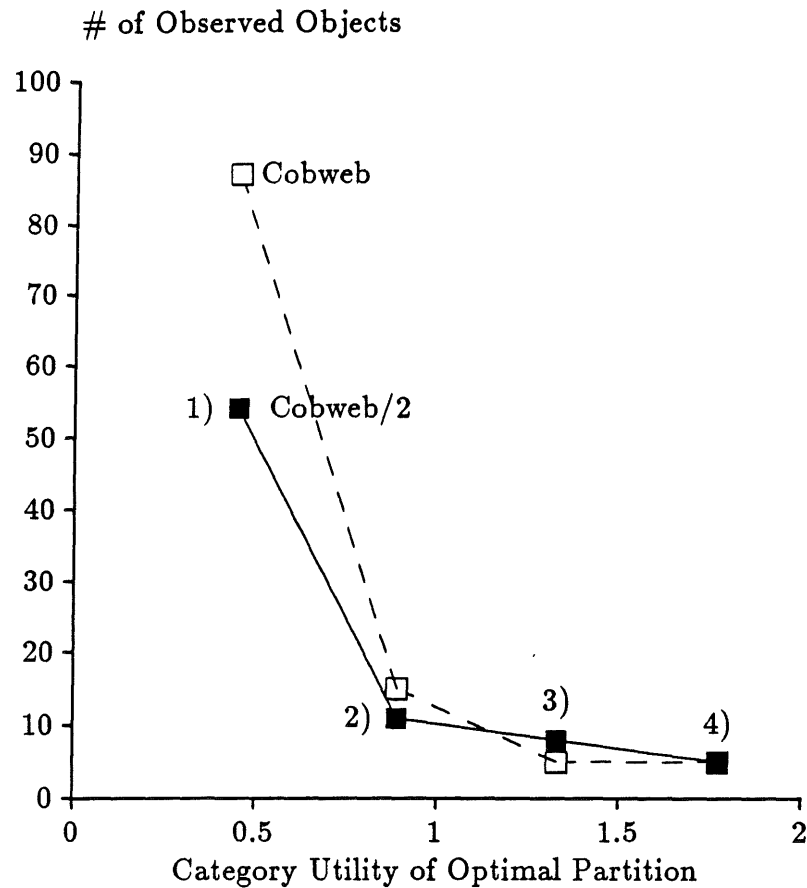


Figure 92

Number of objects required to converge by COBWEB/2

tree classified the objects in the same manner as the optimal partition as a whole, the two partitions were regarded as equivalent.

Figure 92 indicates that as the quality of the optimal partition grows, there is a corresponding drop in the number of objects required by COBWEB/2 to converge on the optimal partition. In fact, on this dimension COBWEB/2 appears to outperform COBWEB, more quickly achieving the optimal partition in domains 3 and 4. Ironically, the reasons for good performance are related to the reasons for poor performance in the last section; indexing more quickly focuses attention on certain attributes as important during classification. In cases, where indices correspond to values that define the optimal partitioning, COBWEB/2 can be expected to

converge on that partition sooner. The caveat is that while COBWEB/2 makes a commitment more quickly as to what attributes are important, it is more likely than COBWEB to be wrong.

8.3.4 Summary

COBWEB/2 has been characterized as an incremental clustering system along three dimensions. In general, along two of these dimensions COBWEB/2 compares favorably with COBWEB. Indexing can be used to reduce the cost of object classification. Furthermore, the discrete nature of indices emphasizes the importance of certain attribute values early on, thus shortening convergence time. However, early in the clustering process, an experiment in the congressional domain indicated that index update may be expensive. Moreover, in domains where a number of orthogonal, competing patterns exist in the data, indices can prematurely designate the 'wrong' attributes as important. Thus, cost and partition optimality can suffer in the early stages of learning with indices. A tentative remedy for these problems is to loosen the requirements for directing indices to nodes. In the most extreme case, indices need not be used at all early in clustering. The COBWEB procedure of comparing an object with respect each child is functionally equivalent to making sure all children are initially indexed. Indices are introduced after a number of objects have primed memory.

8.4 COBWEB/2 Classification Trees

The previous section identified problems with using indices too early in concept formation. It motivated a modification of COBWEB/2 that shifted to indexing only after a number of objects had been added to memory via COBWEB's more conservative incorporation procedures. This section concentrates on the worst-case performance of COBWEB/2 (without priming) in the congressional voting domain. Comparisons between COBWEB/2's output and COBWEB's are also made.

N_1 ('conservative')	N_{51} ('liberal')
$P(A_i = V_{ij} N_1), P(N_1 A_i = V_{ij})$	$P(A_i = V_{ij} N_{51}), P(N_{51} A_i = V_{ij})$
Budget Cuts - yes (0.95,0.83)	Budget Cuts - no (0.80,0.93)
SDI reduction - no (0.89,0.92)	SDI reduction - yes (0.88,0.86)
Contra Aid - yes (0.86,0.87)	
Line-Item Veto - yes (0.88,0.91)	Line-Item Veto - no (0.86,0.84)
MX Production - yes (0.92,0.89)	MX Production - no (0.85,0.91)
Guest Workers - yes (0.81,0.78)	Guest Workers - no (0.78,0.74)
Farm Bill - yes (0.80,0.83)	Farm Bill - no (0.74,0.70)

Table 34

Norms for congressional classes formed without priming

Like COBWEB, COBWEB/2 grouped congressmen into classes corresponding roughly to 'conservatives' and 'liberals'. Classes at the top level of the COBWEB/2 tree closely matched those formed by COBWEB. For example, one node of the COBWEB/2 tree contained all the 'liberals' from the COBWEB tree, except one; this node contained three senators not in the COBWEB 'liberal' cluster. The normative values of COBWEB/2's 'liberal' and 'conservative' clusters are given in Table 34.

While COBWEB/2 grouped senators at the top level into intuitively appealing classes, its decomposition of senators at lower levels was less pleasing. At the second level of the tree there tended to be several singletons, or otherwise small classes. Lower levels did not seem to capture subpatterns (e.g., 'southern-democrats') within the larger patterns represented by 'liberals' and 'conservatives'. Intuitively, this can be explained by the same principles that caused COBWEB/2 to do poorly when confronted with competing patterns in data. COBWEB/2 did fine at discovering the obvious patterns of 'liberal' and 'conservative' voting records, but when confronted with less clear-cut alternatives at lower levels of the tree, it does not fare so well.

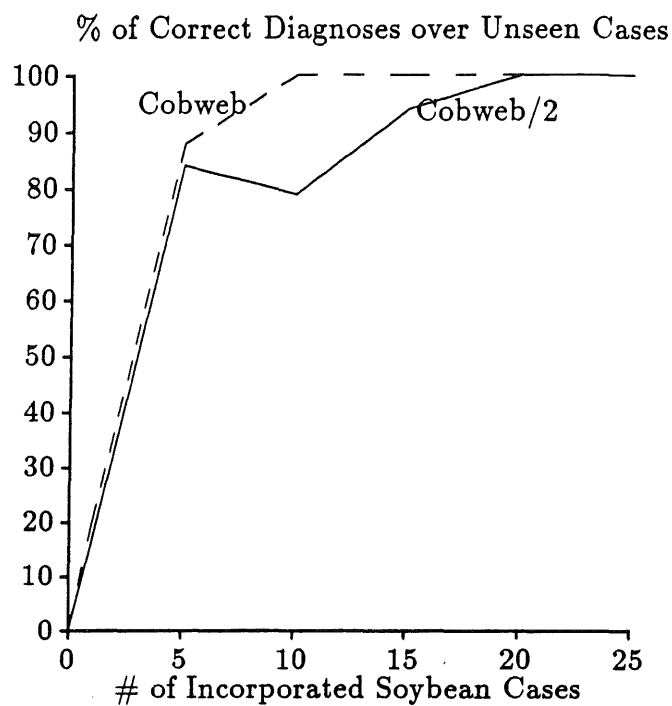


Figure 93

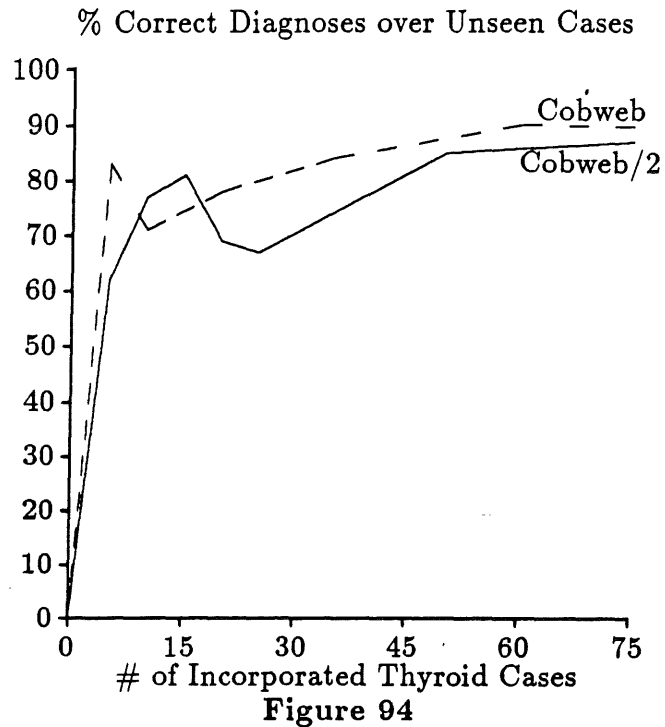
Diagnostic success on Soybean Cases without Priming

8.5 The Utility of COBWEB/2 Classification Trees for Inference

This section characterizes the effectiveness of classification trees formed by COBWEB/2 for inference. While the analysis of COBWEB/2 trees is not nearly as extensive as for COBWEB trees, it shows that COBWEB/2 approximates COBWEB's abilities on this dimension.

COBWEB/2's success at diagnosing soybean disease is given in Figure 93. Without priming, COBWEB/2 requires about 10 more soybean case histories to achieve perfect prediction over unseen cases than COBWEB. A similar effect is shown in Figure 94 for the thyroid domain where COBWEB/2 (without priming) requires more objects to achieve roughly the same accuracy as COBWEB.

While COBWEB/2 takes somewhat longer to reach the levels of diagnostic correctness achieved by COBWEB, it nonetheless reaches these levels with respect to the Diagnostic-condition attribute. With respect to all attributes, COBWEB/2



Diagnostic success on thyroid Cases without Priming

does not fair as well as COBWEB, particularly with respect to attributes of intermediate dependence. However, Figure 95 indicates that it still does fairly well. This graph is reminiscent of the corresponding one for COBWEB (Figure 45).

8.6 Chapter Summary

The goal of this chapter was to test the computational limits of chapter 7's indexing scheme in the context of learning. Such a speculative analysis can indirectly impact claims about the psychological plausibility of a computational model. The main findings of this chapter were that indexing supports cost effective learning and relatively accurate prediction. However, an important caveat is that the indexing scheme is brittle and expensive to maintain early in the learning process. These results suggest that indexing should be less constrained initially, becoming

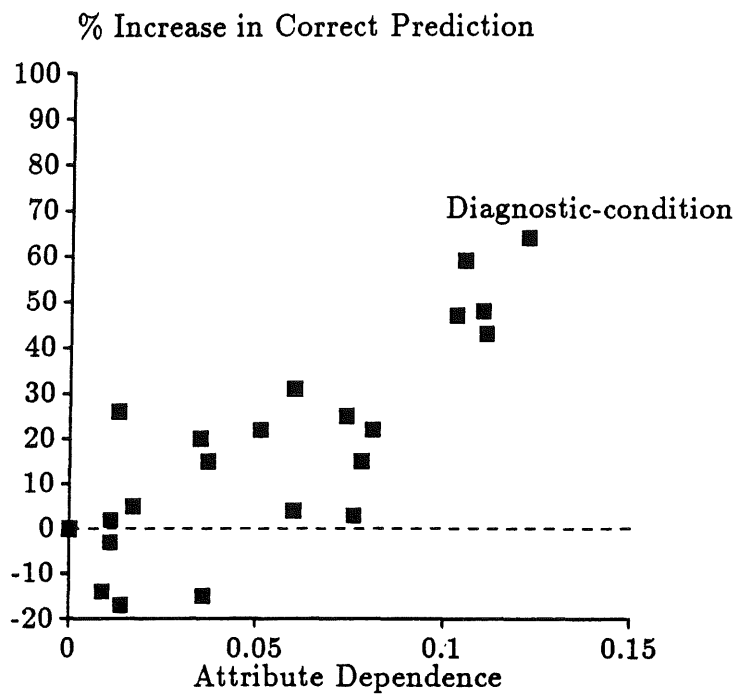


Figure 95

Increase in correct inference using COBWEB/2 without priming

more discretized as learning progresses. While this process has been approximated by using COBWEB's continuous evaluation/incorporation procedures initially, the exact nature of this evolution is left as future work. Importantly, there is little work in cognitive psychology on the emergence of basic level and typicality effects *during* learning that can guide a cognitive model of this process.

CHAPTER 9

Conclusions

9.1 Contributions of the Dissertation

A major bias behind this research effort has been that artificial intelligence and cognitive psychology are cooperative; expertise in cognitive psychology is focused on *describing* intelligent (human) behavior, while artificial intelligence is primarily concerned with *modeling* or *explaining* behavior. In the terminology of chapter 1, psychology is concerned with specification and artificial intelligence is concerned with design, or at least that is the prescriptive view of the dissertation.

More specifically, the work reported herein draws from and contributes to three formally disparate literatures: conceptual clustering, incremental concept formation, and cognitive psychology studies of basic level and typicality effects. The former two are subareas of machine learning that are concerned with the mechanisms underlying concept formation. These mechanisms can be usefully constrained by data on human classification. In return, general mechanisms can be molded into models of human behavior. The dissertation reflects a view of this process as one of step-wise refinement; concept representation and quality measures in COBWEB are suggested by work on human classification; these are consistent with computational desires for incremental processing and prediction accuracy. In turn, these latter properties are consistent with much of human learning and memory; from COBWEB, a memory structure that accounts for the more specific behaviors of typicality and basic level preference is derived.

The remainder of this section details the debts and contributions to machine learning and cognitive psychology, and in doing so, indirectly illustrates the interplay between them.

9.1.1 Conceptual Clustering

COBWEB fits the definition of conceptual clustering as laid out by Michalski and Stepp [MICH80, MIC83A]. The system forms classes whose quality is dependent on the quality of their respective summary or concept level descriptions.

COBWEB uses *probabilistic* concepts to describe object classes. Probabilistic descriptions are a major departure from the logical concept representations of much work in AI and machine learning in general, and conceptual clustering in particular.

COBWEB is evaluated in terms of the performance task of predicting unknown attribute values. This performance task generalizes the task of predicting class membership in learning from examples. Previously, no well-specified performance task was associated with conceptual clustering, but the identification of such a task is imperative if the field is to progress.

In contrast to previous systems, COBWEB is an incremental. It incorporates objects as they are encountered, and thus can be more flexibly applied in real world environments.

9.1.2 Incremental Concept Formation

COBWEB's control structure was inspired by earlier work on incremental concept learning [LEBO82, KOL83A, WINS75]. In terms of search, COBWEB uses a hill-climbing control strategy with operators that allow bidirectional mobility.

While earlier systems embody hill-climbing/bidirectional strategies, this dissertation is novel in explicitly characterizing their behavior as such. This view of incremental learning suggests three dimensions for evaluating incremental systems: update cost, concept quality, and convergence time. Characterization along these

dimensions demonstrates that COBWEB is a robust and cost-effective means of building classification hierarchies.

9.1.3 Psychological Studies of Concepts and Classification

COBWEB's adoption of probabilistic concept representations was motivated by the identification of typicality effects in humans. Additionally, the heuristic measure of class quality used by COBWEB was developed as a predictor of preferred concepts in humans [GLUC85].

In response to objections against probabilistic concepts, a number of alternative representations for individual concepts have been proposed (e.g., exemplar and relational cue models). Chapter 3 points out that probabilistic concept trees of the type built by COBWEB capture the same information as these alternative representations. Objections to probabilistic concepts properly motivate looking to larger organizations of concepts, as well as alternative models of individual concepts.

Object classification in COBWEB inspires an indexing scheme that accounts for simple basic level and typicality effects, as well as interactions between the two classes of phenomena. Computer experiments using the indexing scheme suggest directions for further experiments with human subjects. While it is probable that the indexing system of this chapter is not precisely correct, it embodies an important assumption: psychological phenomena, typicality and basic level effects in particular, cannot be explained without regard to a larger memory structure.

9.1.4 Methodological Biases

In addition to substantive contributions to subareas of machine learning and cognitive psychology, this dissertation advances a relatively novel method of validating inductive learning methods. It builds on the methodological biases of

Quinlan [QUIN83, QUIN86] and others [SCH86C, HAMP83], which favors extensive computer experimentation in a variety of domains.

In addition, COBWEB is characterized with respect to a *normalizing* method, in this case the frequency-based method of chapter 4. While the frequency-based method is a 'straw-man' in many respects, few learning systems are compared to such methods, much less alternative learning techniques. 'Straw-men' give an initial indication of when learning is difficult and help to normalize the results and apparent advantages of more complex inductive methods. In addition, comparisons with ID3 roughly upper bound COBWEB's performance and demonstrate room for improvement along some attributes.

Finally, COBWEB is not simply characterized in a number of domains, but a measure for characterizing the domain itself is forwarded. In chapter 5, COBWEB's ability to predict an attribute's value was shown to vary with the dependence of the attribute on other attribute's. While there may be problems with this function as a general predictor of 'domain difficulty' (e.g., representation language dependence), the philosophy that domain characterization must accompany algorithm characterization harkens back to discussions by Simon [SIMO69], but is relatively unique to current experimental practices. Chapter 6 illustrates how artificial domains can be effectively used to demonstrate the range of a system's behavior. While natural domains are alluring, over reliance on them makes it difficult to test the limits of a system's capabilities.

9.2 Future Work

Many directions for future work are suggested by work on COBWEB and COBWEB/2. Proposals can be roughly segregated by subject matter: machine learning, cognitive psychology, and methodological studies.

9.2.1 Proposals in Machine Learning

Undoubtedly, future work will focus on rectifying a number of COBWEB's limitations. One limiting aspect is the object description language: nominal attribute-value pairs. One way to relax this constraint is to allow numeric, continuously-valued attributes. Gennari, Langley, and Fisher [GENN87] report a modification to COBWEB, CLASSIT, that rewards partitions formed around 'dense' value areas of numeric attributes. An alternative approach that has been implemented, but not extensively tested in COBWEB, discretizes continuous attribute domains into ranges based on how well they contribute to higher-order conceptual descriptions. A range of values can then be treated like a nominal value. This approach is similar to that used by Michalski and Stepp [MIC83B].

A second way of relaxing object description constraints is to allow *structured* objects and concepts. Manipulating structured representations is an important prerequisite for applying conceptual clustering methods in sophisticated problem-solving domains [FIS86B]. As CLUSTER/2 and UNIMEM served as precursors to COBWEB, CLUSTER/S [STEP86, STEP84] and RESEARCHER [LEBO86] are likely starting points for work on incremental clustering of structured objects. Work by Levinson [LEVI84] on incrementally discovering patterns in graph structures is also relevant. Work by Vere [VERE77] on 'clustering' relational productions shows how conceptual clustering methods might be applied to operator descriptions.

Finally, future work will also focus on improving COBWEB's hill-climbing search strategy. Experimentation in chapter 6 suggested that COBWEB's lack of heuristic foresight might preclude its discovery of optimal partitions in domains that lacked clear-cut, dominant patterns in the data. Some situations can be imagined that magnify this limitation considerably. For example, the problem of tracking changes in the environment (e.g., the change of seasons) has been studied in the context of learning from examples by Schlimmer and Granger [SCH86B]. In

real-world domains, a concept formation system must be cognizant of changes in the regularity of the environment. Tracking is equivalent to the problem of dealing with extremely skewed data. Under extreme skew a hill-climbing strategy results in a classification whose utility may irreversibly and progressively degrade.

One solution to this problem might involve loosening the constraints in which the *split* operator is applied. Currently, splitting is only applied if it immediately improves partition quality. However, the heuristic for splitting could be localized; its application could be made dependent on such things as whether partition quality was decreasing over a 'window' of recent trials or on whether an individual node was of poor 'quality' (e.g., too few predictable values). Liberalizing the policy for splitting would allow subtrees to be broken apart and rebuilt (in a different form) through merging. In conjunction with liberalizing conditions under which to split, it may be desirable to liberalize conditions under which new classes are created. One effect of this strategy is that higher-level classes will not be formed through merging or classification until some (constant or variable) threshold number of objects are observed.

9.2.2 Proposals in Cognitive Psychology

The indexing scheme of chapter 7 appears consistent with certain typicality and basic level effects. More importantly, it makes a number of predictions about expected effects (e.g., exhaustive retrieval) and the interaction between basic level and typicality phenomena. The predictions made by the computer model are an obvious point of departure for experimentation with human subjects.

This dissertation has foregone the opportunity of studying the emergence of basic level and typicality effects during learning. However, COBWEB and COBWEB/2 offer a unique framework for generating hypotheses about the evolution of these effects. A pragmatic question is "How does the basic level shift as one becomes expert in a domain?" A task in which principles underlying basic

level effects, typicality effects, and memory update might be particularly relevant is *information or document retrieval* [VANR79]. Queries to a database can be viewed as partial object descriptions. Principles that cause typical objects to be recognized most quickly and lead the vast majority of objects to be recognized with respect to the basic level, can undoubtedly be used to guide queries to the most 'relevant' parts of memory. Using queries to 'update' the database can help optimize retrieval for the most frequently made queries.

9.2.3 Methodological and Comparative Studies

COBWEB and COBWEB/2 have been characterized along a number of dimensions (e.g., convergence time, inference improvement). Importantly, these characterizations have carefully avoided statements of superiority over earlier systems. In fact, there is a very good chance that a number of earlier, but comparable concept formation systems are better than COBWEB (and COBWEB/2) along several important dimensions. The contribution of this work is not that COBWEB is a 'better' system, but that its characterization enumerates a number of dimensions along which comparisons can be made to begin with.

COBWEB is an incremental system that uses a search strategy abstracted from incremental systems like UNIMEM and CYRUS. Describing COBWEB in terms of search has motivated an evaluation of its behavior with respect to the cost and quality of learning. While UNIMEM and CYRUS have not been evaluated along the same dimensions as COBWEB, to some degree COBWEB's characterization is probably extensible to them (e.g., logarithmic update cost). However, a major difference between COBWEB and UNIMEM/CYRUS is that these latter systems form overlapping concepts and hierarchies that are not strict trees; there is no notion of a 'best' partition in these systems. However, while UNIMEM/CYRUS heuristics are somewhat different from COBWEB's, a reasonable expectation is that classes formed by an appropriately biased UNIMEM/CYRUS-type system

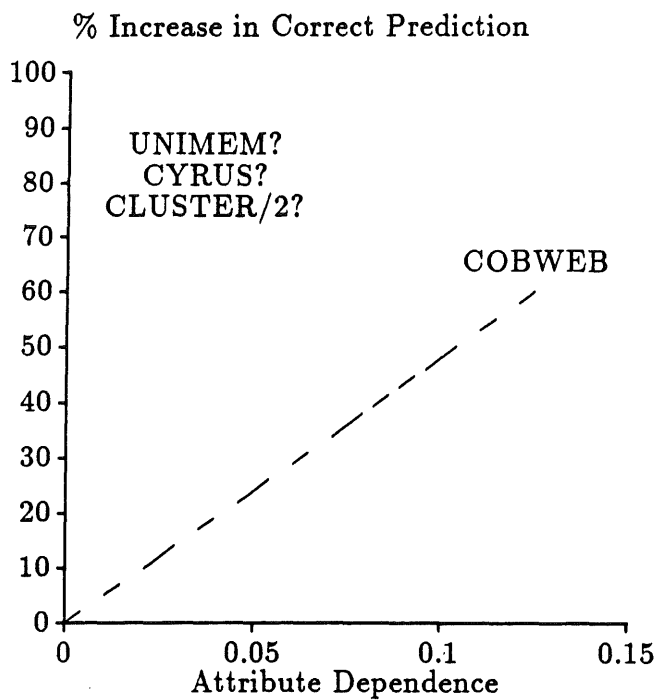


Figure 96

Average increase in correct prediction afforded by COBWEB

would form a superset of the classes formed by COBWEB. In experiments on COBWEB's ability to discover the global optimal partition, UNIMEM/CYRUS might well form classes corresponding to to the globally (as well as the locally) optimal set, even in cases where the global optimal is little 'better' than the local.

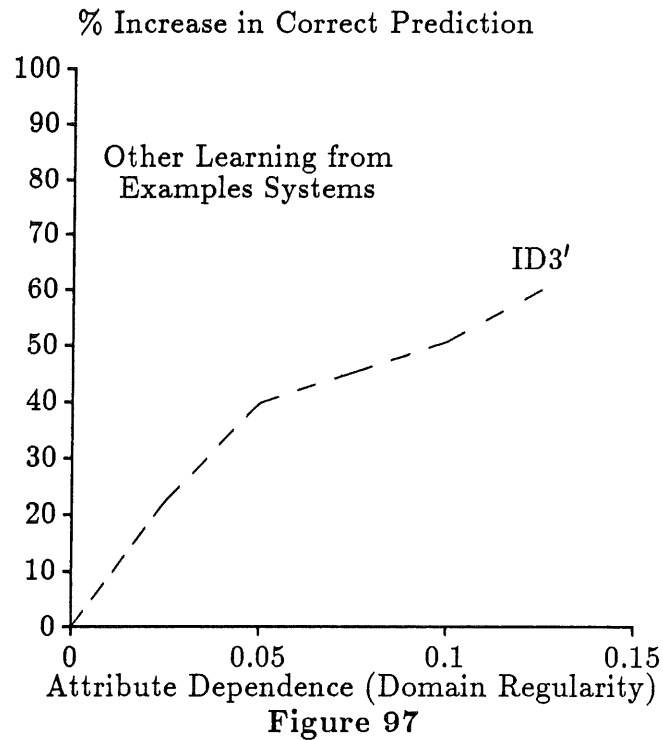
COBWEB classification trees have been characterized by their ability to promote correct prediction of attribute values. In addition, a cheaply computed measure of attribute dependence appears to have some predictive value in telling how well COBWEB's tree will promote prediction. Figure 96 shows the average *increase* afforded by COBWEB classification trees over the frequency-based approach. For example, recall that correct prediction for diagnostic condition in the soybean domain reached 100%, while correctness using the frequency-based approach averaged 36%, leaving a difference of 64%. This data point is represented at the right terminus of the plotted line. An important question asked in Figure 96 is

“How will other methods perform under varying conditions with respect to this task?”

Comparisons with ID3 in chapter 5 indicated that COBWEB matched ID3's correctness with respect to a majority of the attributes. However, ID3 defined a rough upper bound that indicated room for improvement on COBWEB's performance with respect to several (i.e., about 1/5 of the soybean) attributes. An initial hypothesis is that a number of concept formation methods will outperform COBWEB, particularly with respect to attributes of intermediate dependence. UNIMEM/CYRUS form more general hierarchical structures than trees. Chapter 3 pointed out that overlapping concepts may do a better job at capturing important correlations that can aid induction. A nonincremental method like CLUSTER/2 makes a more extensive search of the possible partitionings of objects; it is more likely than COBWEB's hill-climbing strategy to discover the best possible partitions.

While this dissertation has been primarily concerned with concept formation systems, the methodology used to characterize COBWEB can be extended to compare learning from examples systems like ID3 with other systems as implied in Figure 97.⁵³ In the context of learning from examples, the measure of attribute dependence can be regarded as a predictor of 'domain difficulty'. This measure is by no means a perfect predictor and to a significant degree remains untested. However, a more general methodological principle is at issue: a simple statement that algorithm X works in domain Y transmits no information about the difficulty of the domain or the advantages afforded by the algorithm. Domains (natural and artificial) must be characterized before learning algorithms can be characterized. Ideally, a cheaply computed approximation of domain difficulty (e.g., attribute

⁵³ Figures 96 and 97 give only approximate averages of COBWEB and ID3 performance, respectively.



Average increase in correct prediction afforded by ID3 Trees

dependence) should precede and guide the selection of learning strategies under varying conditions [REND87].

9.3 Closing Remarks

This research has tried to embody three themes. First, learning processes in general and conceptual clustering methods in particular must be considered within the larger context of intelligent processing. This has motivated concerns for incremental processing and prediction accuracy. Second, artificial intelligence and cognitive psychology are complementary sciences, but to take advantage of this synergism requires an explicit acknowledgement of the spheres of each field's expertise; psychology is primarily concerned with delimiting intelligent behavior, whereas artificial intelligence is primarily concerned with modeling this behavior. COBWEB draws significantly from studies of basic level and typicality effects, but

hopefully contributes to a better understanding of the mechanisms underlying these effects, as well as pointing out directions for further study. Last, simple case studies may not uncover the strengths and weaknesses of a learning system. Domains must be characterized as well as algorithms. Furthermore, balanced testing in natural and artificial domains can more fully demonstrate the abilities of a system. However, the luxury (and necessity) of this philosophy completely hinges on the more 'constructive' efforts of those researchers to which this dissertation owes much.

REFERENCES

- [ANDE79] ANDERSON, J.R., AND KLINE, P. A learning system and its psychological implications. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, (Tokyo, August 1979), Morgan Kaufmann, Los Altos, CA, 1979, pp. 16-21.
- [BARS84] BARSALOU, W., AND BOWER, G. Discrimination nets as psychological models. *Cognitive Science* 8, 4 (October 1984), 1-26.
- [BRAC85] BRACHMAN, R. J. "I lied about the trees" or, defaults and definitions in knowledge representation. *AI Magazine* 6, 3 (1985), 80-93.
- [BRAD87] BRADSHAW, G. Learning about speech sounds: the NEXUS project. In *Proceedings of the Fourth International Machine Learning Workshop*, (Irvine, CA, June 1987), Morgan Kaufmann, Los Altos, CA, 1987, pp. 1-11.
- [BRUN56] BRUNER, J., GOODNOW, J., AND AUSTIN, G. *A Study of Thinking*, Wiley and Sons, New York, 1956.
- [BUCH69] BUCHANAN, B., SUTHERLAND, G., AND FEIGENBAUM, E. Heuristic DENDRAL: a program for generating explanatory hypotheses in organic chemistry. *Machine Intelligence* 4 (1969), 209-254.
- [BUND84] BUNDY, A., B. BOULAY, HOWE, J., AND PLOTKIN, G. How to get a Ph.D. in AI. In *Artificial Intelligence: Tools, Techniques, and Applications*, T. O'Shea and M. Eisenstadt, Eds., Harper and Row, New York, 1984, pp. 139-154.
- [CARB86] CARBONELL, J., AND HOOD, G. The World Modellers Project: objectives and simulator architecture. In *Machine Learning: A Guide to Current Research*, R. Michalski, J. Carbonell, and T. Mitchell, Eds., Kluwer, Boston, MA, 1986, pp. 29-34.
- [CHAR80] CHARNIAK, E., RIESBECK, C., AND MCDERMOTT, D. *Artificial Intelligence Programming*, LEA, Hillsdale, NJ, 1980.
- [CHAR85] CHARNIAK, E., AND MCDERMOTT, D. *Introduction to Artificial Intelligence*, Addison-Wesley, Reading, MA, 1985.
- [CHEE85] CHEESEMAN, P. In defense of probability. In *Proceedings of the Ninth International Conference on Artificial Intelligence*, (Los Angeles, CA, August 1985), Morgan Kaufmann, Los Alto, CA, 1985, pp. 1002-1009.

- [CHEN85] CHENG, Y., AND FU, K. Conceptual clustering in knowledge organization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 7, 5 (September 1985), 592-598.
- [CLAN84] CLANCEY, W. Classification problem solving. In *Proceedings of the National Conference on Artificial Intelligence*, (Austin, TX, August 1984), Morgan Kaufmann, Los Altos, CA, 1985, pp. 49-55.
- [COLL69] COLLINS, A., AND QUILLIAN, M. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior* 8 (1969), 240-247.
- [COLL75] COLLINS, A., AND LOFTUS, E. A spreading activation theory of semantic processing. *Psychological Review* 82, 6 (1975), 407-428.
- [CONG85] *Congressional Roll Call*, Congressional Quarterly Inc., Washington, D.C., 1985.
- [DIET82] DIETTERICH, T. Learning and inductive inference. In *The Handbook of Artificial Intelligence*, P. Cohen and E. Feigenbaum, Eds., William Kaufmann, Los Altos, CA, 1982, pp. 323-512.
- [DUDA73] DUDA, R. O., AND HART, P. E. *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York, 1973.
- [DUDA79] DUDA, R., GASCHNIG, J., AND HART, P. Model design in the Prospector consultant system for mineral exploration. In *Expert Systems in the Micro Electronic Age*, D. Michie, Ed., Edinburgh University Press, Edinburgh, 1979.
- [ETHE83] ETHERINGTON, D., AND REITER, R. On inheritance hierarchies with exceptions. In *Proceedings of the Third National Conference on Artificial Intelligence*, (Washington, D.C., August 1983), Morgan Kaufmann, Los Altos, CA, 1983, pp. 104-108.
- [EVER80] EVERITT, B. *Cluster Analysis*, Heinemann, London, 1980.
- [FEIG63] FEIGENBAUM, E. A. The simulation of verbal learning behavior. In *Computers and Thought*, E.A. Feigenbaum and J. Feldman, Eds., McGraw-Hill, New York, 1963, pp. 297-309.
- [FEIG84] FEIGENBAUM, E., AND SIMON, H. EPAM-like models of recognition and learning. *Cognitive Science* 8, 4 (1984), 305-336.

- [FIS85A] FISHER, D., AND LANGLEY, P. Approaches to conceptual clustering. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, (Los Angeles, CA, August 1985), Morgan Kaufmann, Los Altos, CA, 1985, pp. 691-697.
- [FIS85B] FISHER, D. A hierarchical conceptual clustering algorithm. Technical Report No. 85-21. Department of Information and Computer Science, University of California, Irvine (April 1985).
- [FIS86A] FISHER, D., AND LANGLEY, P. Methods of conceptual clustering and their relation to numerical taxonomy. In *Artificial Intelligence and Statistics*, W. Gale, Ed., 1986, pp. 77-113.
- [FIS86B] FISHER, D. A proposed method of conceptual clustering for structured and decomposable objects. In *Machine Learning: A Guide to Current Research*, T. Mitchell, J. Carbonell, and R. Michalski, Eds., Kluwer, Boston, MA, 1986, pp. 67-70.
- [FU85] FU, L., AND BUCHANAN, B. Learning intermediate concepts in constructing a hierarchical knowledge base. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, (Los Angeles, CA, August 1985), Morgan Kaufmann, Los Altos, CA, 1985, pp. 659-666.
- [GENN87] GENNARI, J., LANGLEY, P., AND FISHER, D. Models of incremental concept formation. Unpublished manuscript. Department of Information and Computer Science, University of California, Irvine (1987).
- [GLUC85] GLUCK, M., AND CORTER, J. Information, uncertainty, and the utility of categories. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, (Irvine, CA, August 1985), Academic Press, Orlando, FL, 1985, pp. 283-287.
- [HALL85] HALL, R., AND KIBLER, D. Differing methodological perspectives in artificial intelligence. *AI Magazine* 6, 3 (1985), 166-179.
- [HAMP79] HAMPTON, J. Polymorphous concepts in semantic memory. *Journal of Verbal Learning and Behavior* 18, 4 (1979), 441-461.
- [HAMP83] HAMPSON, S. *A Neural Model of Adaptive Behavior*, Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine, 1983.

- [HANS86] HANSON, S., AND BAUER, M. Machine learning, clustering and polymorphy. In *Uncertainty in Artificial Intelligence*, L. Kanal and D. Lemmer, Eds., North-Holland, New York, 1986.
- [HART68] HART, P., NILSSON, N., AND RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics SSC-4* (1968), 100-107.
- [HAYE77] HAYES-ROTH, B., AND HAYES-ROTH, F. Concept learning and the recognition and classification of exemplars. *Journal of Verbal Learning and Verbal Behavior* 16, 3 (June 1977), 321-338.
- [HAYE78] HAYES-ROTH, F., AND McDERMOTT, J. An interference matching technique for inducing abstractions. *Communications of the ACM* 21 (1978), 401-410.
- [HOFF83] HOFFMAN, J., AND ZIESSLER, C. Objectidentifikation in kunstlichen begriffshierarchien. *Zeitschrift fur Psychologie* 194 (1983), 135-167.
- [HUNT66] HUNT, E., MARIN, J., AND STONE, P. *Experiments in Induction*, Academic Press, New York, 1966.
- [JOLI84] JOLICOEUR, P., GLUCK, M., AND KOSSLYN, S. Pictures and names: making the connection. *Cognitive Psychology* 16 (1984), 243-275.
- [JONE83] JONES, G. Identifying basic categories. *Psychological Bulletin* 94, 3 (1983), 423-428.
- [JONE86] JONES, R. Generating predictions to aid the scientific discovery process. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, (Philadelphia, PA, August 1986), Morgan Kaufmann, Los Altos, CA, 1986, pp. 513-517.
- [KIBL87] KIBLER, D., AND AHA, D. Learning representative exemplars of concepts: an initial case study. In *Proceedings of the Fourth International Workshop on Machine Learning*, (Irvine, CA, June 1987), Morgan Kaufmann, Los Altos, CA, 1987, pp. 24-30.
- [KOL83A] KOLODNER, J. Maintaining organization in a dynamic long-term memory. *Cognitive Science* 7 (1983), 243-280.
- [KOL83B] KOLODNER, J. Reconstructive memory: a computer model. *Cognitive Science* 7 (1983), 281-328.

- [KOL83C] KOLODNER, J. Towards an understanding of the role of experience in the evolution from novice to expert. *International Journal of Man-Machine Studies* 9 (1983), 497-518.
- [KOL85A] KOLODNER, J. Experiential processing in natural problem solving. Technical report GIT-ICS-85/23. School of Information and Computer Science, Georgia Institute of Technology, Atlanta (1985).
- [KOL85B] KOLODNER, J. AND KOLODNER, R. *Using Experience in Clinical Problem Solving*, Technical Report GIT-ICS-85/21, School of Information and Computer Science, Georgia Institute of Technology, 1985.
- [KOL87] KOLODNER, J. L. Extending problem solver capabilities through case-based inference. In *Proceedings of the Fourth International Workshop on Machine Learning*, (Irvine, CA, June 1987), Morgan Kaufmann, Los Altos, CA, 1987, pp. 167-178.
- [LANG84] LANGLEY, P., AND SAGE, S. Conceptual clustering as discrimination learning. In *Proceedings of the Fifth Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, (London, Canada, July 1984), 1984, pp. 95-98.
- [LANG85] LANGLEY, P. Discovering qualitative empirical laws. Technical Report 85-18. Department of Information and Computer Science, University of California, Irvine (1985).
- [LAN86A] LANGLEY, P., KIBLER, D., AND GRANGER, R. Components of learning in a reactive environment. In *Machine Learning: A Guide to Current Research*, R. Michalski, J. Carbonell, and T. Mitchell, Eds., Kluwer, Boston, MA, 1986, pp. 167-171.
- [LAN86B] LANGLEY, P., ZYTKOW, J., SIMON, H., AND BRADSHAW, G. The search for regularity: four aspects of scientific discovery. In *Machine Learning, Vol. 2*, R. Michalski, J. Carbonell, and T. Mitchell, Eds., Morgan Kaufmann, Los Altos, CA, 1986, pp. 425-469.
- [LAN87A] LANGLEY, P. A general theory of discrimination learning. In *Production System Models of Learning and Development*, D. Klahr, P. Langley, and R. Neches, Eds., MIT Press, Cambridge, MA, 1987, pp. 99-162.
- [LAN87B] LANGLEY, P., GENNARI, J., AND IBA, W. Hill-climbing theories of learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, (Irvine, CA, June 1987), Morgan Kaufmann, Los Altos, CA, 1987, pp. 312-323.

- [LEB082] LEBOWITZ, M. Correcting erroneous generalizations. *Cognition and Brain Theory* 5, 4 (1982), 367-381.
- [LEB83A] LEBOWITZ, M. Concept learning in a rich input domain. In *Proceedings of the Machine Learning Workshop*, (Urbana-Champaign, IL, June 1983), 1983, pp. 177-182.
- [LEB83B] LEBOWITZ, M. Generalization from natural language text. *Cognitive Science* 7, 1 (1983), 1-40.
- [LEB085] LEBOWITZ, M. Categorizing numeric information for generalization. *Cognitive Science* 9, 3 (1985), 285-308.
- [LEB086] LEBOWITZ, M. Concept learning in a rich input domain: generalization-based memory. In *Machine Learning, v. 2*, R. Michalski, J. Carbonell, and T. Mitchell, Eds., Morgan Kaufmann, Los Altos, CA, 1986, pp. 193-214.
- [LEB087] LEBOWITZ, M. Experiments with incremental concept formation: UNIMEM. *Machine Learning* 2, 2 (1987), 103-138.
- [LEVI84] LEVINSON, R. A self-organizing retrieval system for graphs. In *Proceedings of the National Conference on Artificial Intelligence*, (Austin, TX, August 1984), Morgan Kaufmann, Los Altos, CA, 1984, pp. 203-206.
- [MARR82] MARR, D. *Vision*, W.H. Freeman, San Francisco, 1982.
- [McCA78] McCAULEY, C., AND STITT, C. An individual and quantitative measure of stereotypes. *Journal of Personality and Social Psychology* 36, 9 (September 1978), 929-940.
- [McCA80] McCAULEY, C., STITT, C., AND SEGAL, M. Stereotyping: from prejudice to prediction. *Psychological Bulletin* 87, 1 (January 1980), 195-208.
- [MEDI78] MEDIN, D., AND SCHAFFER, M. A context theory of classification learning. *Psychological Review* 85, 3 (1978), 207-238.
- [MEDI83] MEDIN, D. Structural principles of categorization. In *Perception, Cognition, and Development*, T. Tighe and B. Shepp, Eds., Lawrence Erlbaum, Hillsdale, NJ, 1983, pp. 203-230.
- [MED86A] MEDIN, D., WATTENMAKER, W., AND MICHALSKI, R. Constraints and Preferences in Inductive Learning. Technical Report ISG 86-1. Department of Computer Science, University of Illinois at Urbana-Champaign (1986).

- [MED86B] MEDIN, D., WATTENMAKER, W., AND HAMPSON, S. Family resemblance, conceptual cohesiveness, and category construction. *Cognitive Psychology* 19 (1986), 480-518.
- [MEND79] MENDELSON, E. *Introduction to Mathematical Logic*, Van Nostrand, New York, 1979.
- [MERV80] MERVIS, C. Category structure and the development of categorization. In *Theoretical Issues in Reading Comprehension*, R. Spiro, B. Bruce, and W. F. Brewer, Eds., Lawrence Erlbaum, Hillsdale, NJ, 1980, pp. 279-308.
- [MERV81] MERVIS, C., AND ROSCH, E. Categorization of natural objects. *Annual Review of Psychology* 32 (1981), 89-115.
- [MICH80] MICHALSKI, R. Knowledge acquisition through conceptual clustering: a theoretical framework and algorithm for partitioning data into conjunctive Concepts. *International Journal of Policy Analysis and Information Systems* 4, 3 (1980), 219-243.
- [MICH81] MICHALSKI, R., AND CHILAUSSKY, R. Knowledge acquisition by encoding expert rules versus computer induction by examples. In *Fuzzy Reasoning and its Applications*, B. Gaines, Ed., Academic Press, New York, 1981.
- [MIC83A] MICHALSKI, R., AND STEPP, R. Automated construction of classifications: conceptual clustering versus numerical taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 5, 4 (July 1983), 396-409.
- [MIC83B] MICHALSKI, R. AND STEPP, R. Learning from observation: conceptual clustering. In *Machine Learning*, R. Michalski, J. Carbonell, and T. Mitchell, Eds., Tioga, Palo Alto, CA, Vol. 1, 1983, pp. 331-363.
- [MIC83C] MICHALSKI, R. A theory and methodology of inductive learning. In *Machine Learning*, R. Michalski, J. Carbonell, and T. Mitchell, Eds., Tioga, Palo Alto, CA, Vol. 1, 1983, pp. 83-129.
- [MICH87] MICHALSKI, R. S. How to learn imprecise concepts. In *Proceedings of the Fourth International Workshop on Machine Learning*, (Irvine, CA, June 1987), Morgan Kaufmann, Los Altos, CA, 1987, pp. 50-58.
- [MITC77] MITCHELL, T. Version spaces: a candidate elimination approach to rule learning. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, (Cambridge, MA, August 1977), Morgan Kaufmann, Los Altos, CA, 1977, pp. 305-310.

- [MITC82] MITCHELL, T. Generalization as search. *Artificial Intelligence* 18, 2 (1982), 203-226.
- [MITC83] MITCHELL, T., UTGOFF, P., AND BANERJI, R. Learning by experimentation: acquiring and refining problem-solving heuristics. In *Machine Learning*, R. Michalski, J. Carbonell, and T. Mitchell, Eds., Tioga, Palo Alto, CA, Vol. 1, 1983.
- [MURP82] MURPHY, G. L., AND SMITH, E. E. Basic level superiority in picture categorization. *Journal of Verbal Learning and Verbal Behavior* 21 (1982), 1-20.
- [NEWE63] NEWELL, A., AND SIMON, H. GPS, a program that simulates human thought. In *Computers and Thought*, E. Feigenbaum and J. Feldman, Eds., McGraw Hill, New York, 1963, pp. 279-296.
- [NEWE73] NEWELL, A. Artificial intelligence and the concept of mind. In *Computer Models of Thought and Language*, R. Schank, and K. Colby, Eds., W.H. Freeman, San Francisco, 1973, pp. 1-60.
- [NILS65] NILSSON, N. *Learning Machines*, McGraw Hill, New York, 1965.
- [NILS80] NILSSON, N. *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA, 1980.
- [NORD86] NORDHAUSEN, B. Conceptual clustering using relational information. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, (Philadelphia, PA, August 1986), Morgan Kaufmann, Los Altos, CA, 1986, pp. 508-512.
- [OHL83] OHLSSON, S. Tell me your problems: a psychologist visits AAAI-82. *AISBQ* 46 (1983), 27-30.
- [PAGE80] PAGE-JONES, M. *The Practical Guide to Structured Systems Design*, Yourdon Press, New York, 1980.
- [PEAR85] PEARL, J. Learning hidden causes from empirical data. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, (Los Angeles, CA, August 1985), Morgan Kaufmann, Los Altos, CA, 1985, pp. 567-572.
- [PORT84] PORTER, B., AND KIBLER, D. Learning operator models. In *Proceedings of the Conference of the American Association for Artificial Intelligence*, (Austin, TX, August 1984), Morgan Kaufmann, Los Altos, CA, 1984, pp. 278-282.

- [QUIL68] QUILLIAN, R. Semantic memory. In *Semantic Information Processing*, M. Minsky, Ed., MIT Press, Cambridge, Mass., 1968.
- [QUIN83] QUINLAN, J. R. Learning efficient classification procedures and their application to chess end games. In *Machine Learning*, R. Michalski, J. Carbonell, and T. Mitchell, Eds., Tioga, Palo Alto, CA, Vol. 1, 1983, pp. 463-482.
- [QUIN86] QUINLAN, J. R. Induction of decision trees. *Machine Learning* 1, 1 (January 1986), 81-106.
- [QUI87A] QUINLAN, J. R. Simplifying decision trees. *International Journal of Man-Machine Studies* (to appear).
- [QUI87B] QUINLAN, J. R. Decision trees as probabilistic classifiers. In *Proceedings of the Fourth International Workshop on Machine Learning*, (Irvine, CA, June 1987), Morgan Kaufmann, Los Altos, CA, 1987, pp. 31-37.
- [REED72] REED, S. Pattern recognition and categorization. *Cognitive Psychology* 9 (1972), 382-407.
- [REIN85] REINKE, R., AND MICHALSKI, R. Incremental learning of concept descriptions. In *Machine Intelligence*, J. Hayes, D. Mitchie, and J. Richards, Eds., Vol. 11, 1985.
- [REIT80] REITER, R. A logic for default reasoning. *Artificial Intelligence* 13 (1980), 81-132.
- [REND86] RENDELL, L. A general framework for induction and a study of selective induction. *Machine Learning* 1, 2 (April 1986), 177-226.
- [REND87] RENDELL, L., SESHU, R., AND TCHENG, D. More robust concept learning using dynamically-variable bias. In *Proceedings of the Fourth International Workshop on Machine Learning*, (Irvine, CA, June 1987), Morgan Kaufmann, Los Altos, CA, pp. 1987.
- [RICH83] RICH, E. *Artificial Intelligence*, McGraw Hill, New York, 1983.
- [RIPS73] RIPS, L., SHOBEN, E., AND SMITH, E. Semantic distance and the verification of semantic relations. *Journal of Verbal Learning and Verbal Behavior* 12 (1973), 1-20.
- [ROS75A] ROSCH, E. Cognitive representations of semantic categories. *Journal of Experimental Psychology: General* 104, 3 (1975), 192-233.

- [Ros75B] ROSCH, E., AND MERVIS, C. Family resemblances: studies in the internal structure of categories. *Cognitive Psychology* 7 (1975), 573-605.
- [Ros76A] ROSCH, E., MERVIS, C., GRAY, W., JOHNSON, D., AND BOYES-BRAEM, P. Basic objects in natural categories. *Cognitive Psychology* 8 (1976), 382-439.
- [Ros76B] ROSCH, E., SIMPSON, C., AND MILLER, R. Structural bases of typicality effects. *Journal of Experimental Psychology: Human Perception and Performance* 2 (1976), 491-502.
- [Ros78] ROSCH, E. Principles of categorization. In *Cognition and Categorization*, E. Rosch and B. Lloyd, Eds., Lawrence Erlbaum, Hillsdale, NJ, 1978, pp. 28-49.
- [ROSE58] ROSENBLATT, F. The Perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review* 65 (1958), 386-408.
- [SALZ86] SALZBERG, S. Pinpointing good hypotheses with heuristics. In *Artificial Intelligence and Statistics*, W. Gale, Ed., Addison-Wesley, Reading, MA, 1986, pp. 133-158.
- [SAMM86] SAMMUT, C., AND HUME, D. Learning concepts in a complex robot world. In *Machine Learning: A Guide to Current Research*, R. Michalski, J. Carbonell, and T. Mitchell, Eds., Kluwer, Boston, MA, 1986, pp. 291-294.
- [SCHA77] SCHANK, R., AND ABELSON, R. *Scripts, Plans, Goals, and Understanding*, Lawrence Erlbaum, Hillsdale, NJ, 1977.
- [SCHA82] SCHANK, R. *Dynamic Memory*, Cambridge University Press, New York, 1982.
- [SCH86A] SCHLIMMER, J., AND FISHER, D. A case study of incremental concept induction. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, (Philadelphia, PA, August 1986), Morgan Kaufmann, Los Altos, CA, 1986, pp. 496-501.
- [SCH86B] SCHLIMMER, J., AND GRANGER, R. Beyond incremental processing: tracking concept drift. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, (Philadelphia, PA, August 1986), Morgan Kaufmann, Los Altos, CA, 1986, pp. 502-507.

- [SCH86C] SCHLIMMER, J., AND GRANGER, R. Incremental learning from noisy data. *Machine Learning* 1, 3 (July 1986), 317-354.
- [SIMO69] SIMON, H. A. *Sciences of the Artificial*, MIT Press, Cambridge, MA, 1969.
- [SKYR75] SKYRMS, B. *Choice and Chance*, Wadsworth, Belmont, CA, 1975.
- [SMIT74] SMITH, E., SHOEN, E., AND RIPS, L. Structure and processes in semantic memory: a featural model for semantic decisions. *Psychological Review* 81, 3 (1974), 214-241.
- [SMIT81] SMITH, E., AND MEDIN, D. *Categories and Concepts*, Harvard University Press, Cambridge, MA, 1981.
- [STAN80] STANDISH, T. A. *Data Structure Techniques*, Addison-Wesley, Reading, MA, 1980.
- [STEP84] STEPP, R. *Conjunctive Conceptual Clustering: A Methodology and Experimentation*, Doctoral Dissertation, Department of Computer Science, University of Illinois, Urbana-Champaign, IL, 1984.
- [STEP86] STEPP, R., AND MICHALSKI, R. Conceptual clustering: inventing goal-oriented classifications of structured objects. In *Machine Learning*, Morgan Kaufmann, Los Altos, CA, Vol. 2, 1986, pp. 471-498.
- [VANR79] VAN RIJSBERGEN, C. *Information Retrieval*, Butterworths, Boston, MA, 1979.
- [VERE77] VERE, S. Induction of relational productions in the presence of background information. In *Proceedings of the International Joint Conference on Artificial Intelligence*, (Cambridge, MA, August 1977), Morgan Kaufmann, Los Altos, CA, 1977, pp. 349-355.
- [VERE80] VERE, S. Multilevel counterfactuals for generalization of relational concepts and productions. *Artificial Intelligence* 14, 2 (1980), 139-164.
- [WINS75] WINSTON, P. Learning structural descriptions from examples. In *The Psychology of Computer Vision*, P. Winston, Ed., McGraw Hill, New York, 1975.
- [WOLF80] WOLFF, J. Data compression, generalisation, and overgeneralisation in an evolving theory of language development. In *Proceedings of the AISB-80 Conference on Artificial Intelligence*, 1980, pp. 1-10.

Appendix A

Data for Computer Experiments

Computer experiments in chapters 5 and 9 used three main sets of data: soybean case histories, thyroid case histories, and congressional voting records. The data sets used by COBWEB and COBWEB/2 are given here. In addition, explanations of how the data was transformed from its original source are given.

SOYBEAN CASE HISTORIES

Robert Stepp used a set of 47 soybean case histories to characterize the CLUSTER system in his dissertation. CLUSTER handles both ordinal (linear or integer) and nominal values. While no overt changes were made to the data, COBWEB treated all values as nominal. The domains of the ordinal values were sufficiently small that this appeared a reasonable thing to do. Nonetheless, a loss of information was experienced by this implicit 'transformation' in the data. The 35 attributes, their original type (ordinal or nominal) and their respective domains, are given below.

time-of-occurance (linear): 0:april 1:may 2:june 3:july 4:august
5:september 6:october

plant-stand (nominal): 0:normal 1:less-than-normal

precipitation (linear): 0:below-normal 1:normal 2:above-normal

temperture (linear): 0:below-normal 1:normal 2:above-normal

occurance-of-hail (nominal): 0:no 1:yes

number-years-crop-repeated (linear):
0:none 1:one 2:three 3:four-or-more

damaged-area (nominal): 0:scattered-area 1:low-area 2:upland-area
3:whole-fields

severity (ordinal): 0:minor 1:potentially-severe 2:severe

seed-treatment (nominal): 0:none 1:fungicide 2:other

seed-germination (ordinal): 0:90%-100% 1:80%-89% 2:less-than-80%

plant-height (nominal): 0:normal 1:abnormal

leaf-condition (nominal): 0:normal 1:abnormal

leaf-spots-halos (nominal): 0:absent 1:with-yellow-halos
2:without-yellow-halos

leaf-spots-margin (nominal): 0:water-soaked 1:not-water-soaked
2:not-applicable

size-of-leaf-spots (nominal): 0:less-than-one-eighth-inch
1:greater-than-eighth-inch 2:not-applicable

shot-holing (nominal): 0:absent 1:present

leaf-malformation (nominal):
0:absent 1:present

leaf-mildew-growth (nominal):
0:absent 1:upper-leaf-surface
2:lower-leaf-surface

condition-of-stem (nominal):	0:normal 1:abnormal
stem-lodging (nominal):	0:absent 1:present
stem-cankers (nominal):	0:absent 1:below-soil 2:slightly-above-soil-line 3:above-send-node
canker-lesion-color (nominal):	0:not-applicable 1:brown 2:dark-brown-or-black 3:tan
fruiting-bodies-on-stem (nominal):	0:absent 1:present
outer-stem-decay (nominal):	0:absent 1:firm-and-dry 2:watery-and-soft
mycelium-on-stem (nominal):	0:absent 1:present
internal-discoloration-of-stem (nominal):	0:none 1:brown 2:black
scerotia-internal-or-external (nominal):	0:absent 1:present
fruit-pod-condition (nominal):	0:normal 1:diseased 2:few-or-none 3:not-applicable
fruit-spots (nominal):	0:absent 1:colored-spots 2:brown-spots-black-specks 3:distorted-pods 4:not-applicable
seed-condition (nominal):	0:normal 1:abnormal
seed-mold-growth (nominal):	0:absent 1:present
seed-discoloration (nominal):	0:absent 1:present
seed-size (nominal):	0:normal 1:smaller-than-normal
seed-shriveling (nominal):	0:absent 1:present
root-condition (nominal):	0:normal 1:rotted 2:galls-or-cysts

Four diagnostic conditions were represented in the data: Diaporthe Stem Canker (10 cases), Charcoal Rot (10 cases), Rhizoctonia Root Rot (10 cases), and Phytophthora Rot (17 cases). The case histories are listed below with values for the 35 attributes given in the order listed above. All instances of Diaporthe Stem Canker are listed first, followed by all cases of Charcoal Rot, all cases of Rhizoctonia Root Rot, and finishing with Phytophthora Rot.

Diaporthe Stem Canker

4 0 2 1 1 1 0 1 0 2 1 1 0 2 2 0 0 0 1 0 3 1 1 1 0 0 0 0 4 0 0 0 0 0 0
 5 0 2 1 0 3 1 1 1 2 1 1 0 2 2 0 0 0 1 1 3 0 1 1 0 0 0 0 4 0 0 0 0 0 0
 3 0 2 1 0 2 0 2 1 1 1 1 0 2 2 0 0 0 1 0 3 0 1 1 0 0 0 0 4 0 0 0 0 0 0
 6 0 2 1 0 1 1 1 0 0 1 1 0 2 2 0 0 0 1 1 3 1 1 1 0 0 0 0 4 0 0 0 0 0 0
 4 0 2 1 0 3 0 2 0 2 1 1 0 2 2 0 0 0 1 0 3 1 1 1 0 0 0 0 4 0 0 0 0 0 0
 5 0 2 1 0 2 0 1 1 0 1 1 0 2 2 0 0 0 1 1 3 1 1 1 0 0 0 0 4 0 0 0 0 0 0
 3 0 2 1 0 2 1 1 0 1 1 1 0 2 2 0 0 0 1 1 3 0 1 1 0 0 0 0 4 0 0 0 0 0 0
 3 0 2 1 0 1 0 2 1 2 1 1 0 2 2 0 0 0 1 0 3 0 1 1 0 0 0 0 4 0 0 0 0 0 0
 6 0 2 1 0 3 0 1 1 1 1 1 0 2 2 0 0 0 1 0 3 1 1 1 0 0 0 0 4 0 0 0 0 0 0
 6 0 2 1 0 1 0 1 0 2 1 1 0 2 2 0 0 0 1 0 3 1 1 1 0 0 0 0 4 0 0 0 0 0 0

Charcoal Rot

6 0 0 2 1 0 2 1 0 0 1 1 0 2 2 0 0 0 1 1 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0
 4 0 0 1 0 2 3 1 1 1 1 1 0 2 2 0 0 0 1 0 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0
 5 0 0 2 0 3 2 1 0 2 1 1 0 2 2 0 0 0 1 0 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0
 6 0 0 1 1 3 3 1 1 0 1 1 0 2 2 0 0 0 1 0 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0
 3 0 0 2 1 0 2 1 0 1 1 1 0 2 2 0 0 0 1 0 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0
 4 0 0 1 1 1 3 1 1 1 1 1 0 2 2 0 0 0 1 1 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0
 3 0 0 1 0 1 2 1 0 0 1 1 0 2 2 0 0 0 1 0 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0
 5 0 0 2 1 2 2 1 0 2 1 1 0 2 2 0 0 0 1 1 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0
 6 0 0 2 0 1 3 1 1 0 1 1 0 2 2 0 0 0 1 0 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0
 5 0 0 2 1 3 3 1 1 2 1 1 0 2 2 0 0 0 1 0 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0

Rhizoctonia Root Rot

0 1 2 0 0 1 1 1 1 1 1 0 0 2 2 0 0 0 1 0 1 1 0 1 1 0 0 3 4 0 0 0 0 0 0
 2 1 2 0 0 3 1 2 0 1 1 0 0 2 2 0 0 0 1 0 1 1 0 1 0 0 0 3 4 0 0 0 0 0 0
 2 1 2 0 0 2 1 1 0 2 1 0 0 2 2 0 0 0 1 0 1 1 0 1 1 0 0 3 4 0 0 0 0 0 0
 0 1 2 0 0 0 1 1 1 2 1 0 0 2 2 0 0 0 1 0 1 1 0 1 0 0 0 3 4 0 0 0 0 0 0
 0 1 2 0 0 2 1 1 1 1 1 0 0 2 2 0 0 0 1 0 1 1 0 1 0 0 0 3 4 0 0 0 0 0 0
 4 0 2 0 1 0 1 2 0 2 1 1 0 2 2 0 0 0 1 1 1 1 0 1 1 0 0 3 4 0 0 0 0 0 0
 2 1 2 0 0 3 1 2 0 2 1 0 0 2 2 0 0 0 1 0 1 1 0 1 1 0 0 3 4 0 0 0 0 0 0
 0 1 2 0 0 0 1 1 0 1 1 0 0 2 2 0 0 0 1 0 1 1 0 1 0 0 0 3 4 0 0 0 0 0 1
 3 0 2 0 1 3 1 2 0 1 1 0 0 2 2 0 0 0 1 1 1 1 0 1 1 0 0 3 4 0 0 0 0 0 0
 0 1 2 0 0 1 1 2 1 2 1 0 0 2 2 0 0 0 1 0 1 1 0 1 0 0 0 3 4 0 0 0 0 0 0

Phytophthora Rot

2 1 2 1 1 3 1 2 1 2 1 1 0 2 2 0 0 0 1 0 2 2 0 1 0 0 0 3 4 0 0 0 0 0 1
0 1 1 1 0 1 1 1 0 0 1 1 0 2 2 0 0 0 1 0 1 2 0 0 0 0 0 3 4 0 0 0 0 0 1
3 1 2 0 0 1 1 2 1 0 1 1 0 2 2 0 0 0 1 0 2 2 0 0 0 0 0 3 4 0 0 0 0 0 1
2 1 2 1 1 1 1 2 0 2 1 1 0 2 2 0 0 0 1 0 1 2 0 1 0 0 0 3 4 0 0 0 0 0 1
1 1 2 0 0 3 1 1 1 2 1 1 0 2 2 0 0 0 1 0 2 2 0 0 0 0 0 3 4 0 0 0 0 0 1
1 1 2 1 0 0 1 2 1 1 1 1 0 2 2 0 0 0 1 0 2 2 0 0 0 0 0 3 4 0 0 0 0 0 1
0 1 2 1 0 3 1 1 0 0 1 1 0 2 2 0 0 0 1 0 1 2 0 0 0 0 0 3 4 0 0 0 0 0 1
2 1 2 0 0 1 1 2 0 0 1 1 0 2 2 0 0 0 1 0 1 2 0 0 0 0 0 3 4 0 0 0 0 0 1
3 1 2 0 0 2 1 2 1 1 1 1 0 2 2 0 0 0 1 0 2 2 0 0 0 0 0 3 4 0 0 0 0 0 1
3 1 1 0 0 2 1 2 1 2 1 1 0 2 2 0 0 0 1 0 2 2 0 0 0 0 0 3 4 0 0 0 0 0 1
0 1 2 1 1 1 1 1 0 0 1 1 0 2 2 0 0 0 1 0 1 2 0 1 0 0 0 3 4 0 0 0 0 0 1
1 1 2 1 1 3 1 2 0 1 1 1 0 2 2 0 0 0 1 1 1 2 0 1 0 0 0 3 4 0 0 0 0 0 1
1 1 2 0 0 0 1 2 1 0 1 1 0 2 2 0 0 0 1 0 2 2 0 0 0 0 0 3 4 0 0 0 0 0 1
1 1 2 1 1 2 3 1 1 1 1 1 0 2 2 0 0 0 1 0 2 2 0 1 0 0 0 3 4 0 0 0 0 0 1
2 1 1 0 0 3 1 2 0 2 1 1 0 2 2 0 0 0 1 0 1 2 0 0 0 0 0 3 4 0 0 0 0 0 1
0 1 1 1 1 2 1 2 1 0 1 1 0 2 2 0 0 0 1 1 2 2 0 1 0 0 0 3 4 0 0 0 0 0 1
0 1 2 1 0 3 1 1 0 2 1 1 0 2 2 0 0 0 1 0 1 2 0 0 0 0 0 3 4 0 0 0 0 0 1

THYROID CASE HISTORIES

J. R. Quinlan kindly supplied a number of researchers at UCI with a sizable database of thyroid case histories taken from the Garvan Institute, Australia. A total of 6326 case histories were supplied. Three diagnostic conditions were present in the data: negative, hypothyroid, and sick euthyroid. Each case history was described by at most 25 attributes. Of this original data, 151 records were classified as hypothyroid, 294 were classified as sick euthyroid, and the remainder were negative. These original attributes and their original domains are given below.

Attributes	Domains
age	continuous
sex	Male (M), Female (F)
on thyroxine	false (f), true (t)
query on thyroxine	false (f), true (t)
on antithyroid medication	false (f), true (t)
thyroid surgery	false (f), true (t)
query hypothyroid	false (f), true (t)
query hyperthyroid	false (f), true (t)
pregnant	false (f), true (t)
sick	false (f), true (t)
tumor	false (f), true (t)
lithium	false (f), true (t)
goitre	false (f), true (t)
TSH measured	yes (y), no (n)
TSH	continuous
T3 measured	yes (y), no (n)
T3	continuous
TT4 measured	yes (y), no (n)
TT4	continuous
T4U measured	yes (y), no (n)
T4U	continuous
FTI measured	yes (y), no (n)
FTI	continuous
TBG measured	yes (y), no (n)
TBG	continuous

For the experiments of this dissertation, the data was not used in its raw form. Three major changes were made. First, all continuous attributes were nominalized. This was done using value ranges used by experts in the thyroid field. The ranges representing nominal values are given below.

Attribute	Ranges
age	under 18 (1), 18-29 (2), 30-44 (3), 45-64 (4), over 64 (5)
TSH	under 6 inclusive (normal), over 6 (high)
T3	under 1.2 (low), 1.2-2.8 (normal), over 2.8 (high)
TT4	under 60 (low), 60-150 (normal), over 150 (high)
T4U	under 0.6 (low), 0.6-1.25 (normal), over 1.25 (high)
FTI	under 65 (low), 65-155 (normal), over 155 (high)
TBG	under 12 (low), 12-30 (normal), over 30 (high)

The second major change to the data was the elimination of all attributes of the form, 'X measured' (e.g., TSH measured). These attributes appeared to be redundant when using COBWEB, since COBWEB can incorporate objects with missing information. That is, if TSH is present in an instance then TSH was measured, otherwise it was not measured.

Lastly, rather than using all 6326 instances, 150 instances were used. 50 instances were randomly selected from each diagnostic class (i.e., negative, hypothyroid, sick euthyroid). This makes the task of *increasing* correct prediction of diagnostic condition considerably easier than Quinlan has it, but it provides a good 'base line' attribute for comparing against the frequency based approach. In effect, making diagnostic condition equiprobable raises its attribute dependence score. However, recall that in the thyroid domain (150 case histories) there were many other attributes at the low end of the attribute dependence scale.

The 150 case histories follow – one case history per line. Attribute values are given in the following order with abbreviated value names:

age	1,2,3,4,5
sex	M, F
on thyroxine	f,t
query on thyroxine	f,t
on antithyroid medication	f,t
thyroid surgery	f,t
query hypothyroid	f,t
query hyperthyroid	f,t
pregnant	f,t
sick	f,t
tumor	f,t
lithium	f,t
goitre	f,t
TSH	n,h
T3	l,n,h
TT4	l,n,h
T4U	l,n,h
FTI	l,n,h
TBG	l,n,h

Missing attributes are marked with a '?'. The instances are arranged so that all negative instances come first, then all hypothyroid instances, then all sick euthyroid instances.

negative

3,	F,	f,	f,	f,	t,	t,	f,	f,	f,	f,	f,	h,	n,	n,	n,	n,	?
3,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	?,	n,	n,	n,	?
5,	F,	f,	f,	f,	f,	t,	f,	f,	f,	f,	t,	n,	h,	n,	n,	n,	?
4,	F,	f,	f,	f,	t,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
?,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
4,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
4,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
2,	F,	t,	f,	f,	f,	t,	f,	f,	f,	f,	f,	?,	?,	?,	?,	?,	h
?,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	h,	n,	n,	h,	n,	?
4,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	?,	?,	n,	n,	n,	?
1,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	?,	n,	n,	n,	?
1,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	l,	l,	l,	n,	?
5,	F,	f,	f,	f,	t,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?

4,	M,	f,	f,	f,	f,	f,	f,	f,	t,	f,	f,	n,	n,	n,	n,	h,	?
4,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
3,	F,	f,	f,	t,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
5,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
2,	M,	f,	f,	f,	f,	f,	t,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
5,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
5,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
?,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
2,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	h,	h,	h,	n,	?
4,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
3,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	?,	?,	?,	?,	?,	h
4,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
3,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
4,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
3,	F,	f,	f,	f,	f,	f,	t,	f,	f,	f,	f,	?,	?,	?,	?,	?,	h
4,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	h,	h,	n,	?
3,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
?,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	?,	?,	n,	n,	n,	?
3,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	?,	?,	n,	n,	n,	?
5,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	l,	n,	n,	n,	?
4,	F,	t,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
5,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
3,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
5,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
?,	F,	f,	f,	f,	f,	f,	f,	t,	f,	f,	f,	n,	h,	n,	h,	n,	?
?,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	h,	l,	h,	l,	h,	?
4,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	h,	n,	?
4,	F,	f,	f,	t,	f,	f,	f,	f,	f,	f,	f,	h,	n,	n,	n,	n,	?
3,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	h,	h,	n,	h,	n,	?
2,	F,	f,	f,	f,	f,	f,	f,	f,	t,	f,	f,	n,	h,	n,	n,	n,	?
?,	F,	f,	f,	f,	f,	f,	t,	f,	f,	f,	f,	?,	?,	n,	n,	n,	?
5,	F,	t,	f,	f,	f,	t,	f,	f,	f,	f,	f,	n,	n,	h,	n,	h,	?
5,	F,	f,	f,	f,	f,	t,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
3,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	l,	n,	n,	n,	?
5,	F,	f,	f,	f,	f,	t,	f,	f,	f,	f,	f,	n,	l,	n,	n,	n,	?
3,	F,	f,	f,	f,	f,	t,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
5,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	n,	n,	n,	n,	?
3,	F,	f,	f,	f,	f,	f,	t,	f,	f,	f,	f,	n,	h,	h,	h,	n,	?

hypothyroid

5,	F,	f,	f,	f,	t,	f,	f,	f,	f,	f,	h,	n,	l,	n,	l,	?
2,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	h,	?,	l,	n,	l,	?
2,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	h,	l,	l,	n,	l,	?
5,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	h,	l,	l,	n,	l,	?
5,	F,	t,	f,	f,	f,	f,	f,	f,	f,	f,	h,	?,	l,	n,	l,	?

sick euthyroid

?	M	f	f	f	f	f	f	f	f	f	f	n	l	l	n	n	?
?	M	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
2	F	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
3	M	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
1	M	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
4	M	f	f	f	f	f	f	f	t	f	f	n	l	n	n	n	?
?	M	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
5	F	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
3	M	f	f	f	f	f	f	f	f	f	f	n	n	n	n	n	?
?	M	f	f	f	f	f	f	t	f	f	f	n	l	n	n	n	?
4	F	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
?	F	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
4	F	f	f	f	f	f	f	f	t	f	f	n	l	l	l	l	?
5	M	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
?	F	f	f	f	f	f	f	f	f	f	f	n	n	h	n	n	?
4	F	f	f	f	f	f	f	f	f	f	f	n	h	h	n	h	?
5	M	f	f	f	f	f	f	f	t	f	f	n	l	n	n	n	?
5	F	f	f	f	f	f	f	f	t	f	f	n	l	n	n	n	?
?	F	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
4	M	f	f	f	f	f	f	f	t	f	f	n	l	l	n	l	?
5	M	f	f	f	f	f	f	f	f	f	f	n	l	l	n	n	?
4	M	f	f	f	f	f	f	f	t	f	f	n	l	n	n	n	?
5	F	f	f	f	f	f	f	f	f	t	f	n	l	n	n	n	?
?	F	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
5	M	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
?	?	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
3	F	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
?	F	f	f	f	t	f	f	f	f	f	f	n	l	n	n	n	?
5	F	f	f	f	f	f	f	f	t	f	f	n	l	n	n	n	?
?	?	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
5	F	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
5	F	f	f	f	f	f	f	f	t	f	f	n	l	n	n	n	?
2	M	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
4	M	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
5	F	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
5	F	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
5	F	f	f	f	f	f	f	f	t	f	f	n	l	l	n	n	?
5	M	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
2	M	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
5	M	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
5	M	f	f	f	f	f	f	f	f	f	f	n	l	n	n	h	?
3	F	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?
?	F	f	f	f	f	f	f	f	f	f	f	n	l	n	n	n	?

5,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	l,	n,	n,	n,	?
5,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	l,	n,	n,	n,	?
5,	F,	t,	f,	f,	f,	t,	f,	f,	f,	f,	f,	f,	n,	l,	n,	n,	n,	?
4,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	l,	n,	n,	n,	?
5,	F,	f,	f,	f,	f,	f,	f,	f,	t,	f,	f,	f,	n,	l,	n,	n,	n,	?
4,	F,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	n,	l,	n,	n,	n,	?
5,	M,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	f,	?,	l,	n,	n,	n,	?

SENATE VOTING RECORDS

The votes of 100 senators that were cast along 14 'key votes' were taken from the 1985 edition of the Congressional Quarterly. Each senator was considered an object. Each vote was considered an attribute. In the original data, each vote could have one of eight values. The values were:

Y	(voted yes)	
#	(paired for)	y (for bill)
+	(announced for)	
N	(voted no)	
X	(paired against)	n (against bill)
-	(announced against)	
P	(voted present)	
C	(voted present to avoid conflict of interest)	

The first three values (Y, #, +) are all instances of a senator being for a bill. Thus these values were mapped onto a single value, 'y'. Likewise, the values (N, X, -) were mapped onto 'n'. The last two values were treated as unknown (?), along with cases where there was absolutely no value for a senator along a particular vote.

The vote (y, n, or ?) of senators are listed in the following order:

- Emergency Farm Credit
- MX Missile Production
- Budget Resolution
- Anti-Missile Defense
- Nicaragan 'Contra' Aid
- United Nations Budget
- Gun Control
- Line-Item Veto
- School Prayer
- Seasonal Workers
- Toxic-Waste Victims Aid
- Gramm-Rudman-Hollings
- Textile Import Limits
- Farm Bill

The name of the senator is given first, although neither this or political party were given in the object descriptions given to COBWEB.

Republican

denton	y	y	y	n	y	y	y	y	n	y	y	y	y	y
murkowski	y	y	y	n	y	y	y	y	n	y	y	y	n	y
stevens	y	y	y	n	y	y	y	y	y	y	y	y	n	y
goldwater	?	y	y	n	?	y	y	y	y	y	y	y	n	y
wilson	n	y	y	n	y	y	y	y	y	y	y	y	n	y
armstrong	n	y	y	n	y	?	?	y	n	y	y	y	n	y
weicker	y	n	y	y	n	n	y	n	y	n	y	n	y	y
roth	n	y	y	n	y	y	y	y	n	n	y	y	y	y
hawkins	y	y	n	n	y	y	y	y	n	y	y	y	y	y
mattingly	n	y	y	n	y	y	y	y	n	y	y	y	y	y
mcclure	n	y	y	n	y	y	y	y	n	y	y	y	y	y
symms	n	y	y	n	y	y	y	y	n	y	y	y	n	y
lugar	n	y	y	n	y	y	y	y	y	y	y	y	n	y
quayle	n	y	y	n	y	y	y	y	n	y	y	y	n	y
grassley	y	n	y	y	y	y	y	y	n	n	y	y	n	n
dole	n	y	y	n	y	y	y	y	n	y	y	y	y	y
kassebaum	n	n	y	y	y	y	y	y	y	y	y	n	n	y
mcconnell	n	y	y	n	y	?	y	y	n	y	y	y	y	y
cohen	n	y	y	n	n	?	y	y	y	n	n	y	y	y
mathias	y	y	n	y	n	n	n	n	y	n	n	?	y	y
boschwitz	n	y	y	n	y	y	y	y	y	y	y	y	n	y
durenberger	y	n	y	n	y	y	y	n	y	n	n	y	n	n
cochran	n	y	y	n	y	y	y	y	n	y	y	y	y	y
danforth	y	y	y	n	y	y	y	y	y	n	y	y	n	y
hecht	n	y	y	n	y	y	y	y	n	y	y	y	n	y
laxalt	n	y	y	n	y	?	y	y	n	y	y	y	y	y
humphrey	n	y	y	n	y	y	y	y	n	y	n	y	n	y
rudman	n	y	y	n	y	?	y	y	y	y	y	y	y	y
domenici	n	y	y	n	y	y	y	y	y	y	y	y	y	y
damato	y	y	n	n	y	y	y	y	y	y	n	y	y	y
east	n	y	y	?	y	y	y	y	?	y	?	y	y	?
helms	n	y	y	n	y	y	y	y	n	y	y	y	y	y
andrews	y	n	y	n	y	y	y	n	y	n	?	y	n	y
nickles	n	y	y	n	y	y	y	y	n	y	y	y	n	n
hatfield	y	n	y	y	n	n	y	n	y	y	y	n	n	y
packwood	n	y	y	n	n	y	y	n	y	n	n	y	n	y
heinz	n	y	y	n	y	y	y	y	y	n	n	y	y	y
specter	y	y	n	?	n	y	y	y	y	y	n	y	y	y
chafee	n	y	y	y	n	n	n	y	y	n	n	y	n	y

thurmond	n	y	y	n	y	y	y	y	n	y	y	y	y	y
abdnor	y	y	y	n	y	y	y	y	n	y	y	y	n	n
pressler	y	n	y	n	y	y	y	y	n	n	y	y	n	n
gramm	n	y	y	n	y	y	y	y	n	y	y	y	n	y
garn	n	y	y	n	y	y	y	y	n	y	y	y	y	y
hatch	n	y	y	n	y	y	y	y	y	y	n	y	y	y
stafford	n	n	y	n	n	y	y	n	y	n	n	n	n	?
trible	n	y	y	n	y	y	y	y	n	y	y	y	y	y
warner	n	y	y	n	y	y	y	y	n	y	y	y	y	y
evans	n	y	y	n	n	y	y	y	y	y	y	y	n	y
gorton	n	y	y	n	n	y	y	y	y	y	y	y	n	y
kasten	y	y	y	n	y	?	y	y	n	n	y	y	y	n
simpson	n	y	y	n	y	y	y	y	n	n	y	y	n	y
wallop	?	y	y	n	y	y	y	y	n	y	y	y	n	y

Democrat

proxmire	n	n	n	y	n	y	y	y	y	n	n	y	y	y
rockefeller	y	n	n	y	y	?	y	n	y	n	n	y	y	n
byrd	y	y	n	n	y	y	y	n	y	n	n	n	y	n
leahy	y	n	n	y	n	?	y	y	y	y	n	y	y	n
bentsen	y	y	n	n	y	y	y	n	n	y	y	y	y	y
sasser	y	n	n	y	n	y	y	n	n	y	n	y	y	n
gore	y	y	n	y	n	y	y	n	y	y	n	y	y	n
hollings	y	n	n	n	y	y	y	y	y	y	y	y	y	n
pell	y	n	n	y	n	n	n	y	y	n	n	n	y	n
boren	y	y	n	n	y	y	y	y	y	y	?	y	n	n
metzenbaum	y	n	n	y	n	y	n	n	y	n	n	n	y	n
glenn	y	n	n	n	n	y	y	n	y	n	y	n	y	n
burdick	y	n	n	y	n	y	y	n	y	n	n	y	n	n
moynihan	y	n	n	y	n	n	n	n	y	n	?	n	y	n
bingaman	y	n	n	n	n	y	y	n	y	n	y	n	y	n
lautenberg	y	n	n	y	n	y	n	n	y	n	n	n	y	n
bradley	y	n	n	n	n	?	?	n	y	n	n	n	n	n
zorinsky	y	y	y	n	n	y	y	y	y	n	y	y	n	n
exon	y	n	y	n	y	y	y	y	n	y	n	n	n	n
melcher	y	n	n	y	n	?	y	n	y	n	n	y	y	n
baucus	y	n	n	y	n	y	y	n	y	y	n	y	n	n
eagleton	y	n	n	y	n	y	y	n	y	n	n	n	y	n
stennis	y	y	n	?	y	y	?	n	n	?	?	y	y	n
riegle	y	n	n	y	n	y	y	n	y	y	n	n	y	n
levin	y	n	n	y	n	y	n	n	y	n	n	y	y	n
kerry	y	n	n	y	n	n	n	n	y	n	n	y	y	n
kennedy	y	n	n	y	n	?	n	y	y	n	n	y	y	n
sarbanes	y	n	n	y	n	n	n	n	y	n	n	n	y	n

mitchell	y	n	n	y	n	y	y	n	y	y	n	n	y	n
long	y	y	n	n	y	n	?	n	n	n	y	y	y	y
johnston	y	n	n	y	y	y	y	n	n	n	y	n	y	y
ford	y	n	n	y	y	y	y	n	n	n	n	y	y	n
harkin	y	n	n	y	n	y	y	n	y	n	n	n	n	n
simon	y	n	n	y	n	?	?	n	y	n	n	y	y	n
dixon	y	n	n	n	y	y	y	y	y	n	n	y	y	n
matsunaga	y	n	n	y	n	n	n	n	y	n	n	n	n	y
inouye	y	n	n	y	n	n	n	n	y	?	n	n	n	?
nunn	y	y	n	n	y	y	y	y	y	y	y	y	y	y
chiles	y	n	n	y	y	y	y	n	y	n	n	n	?	y
biden	y	n	n	y	n	n	y	y	y	n	n	y	y	n
dodd	y	n	n	y	n	y	n	n	y	n	n	y	y	n
hart	y	n	n	y	n	n	n	n	y	n	n	n	n	n
cranston	y	n	n	y	n	?	n	n	y	n	n	n	n	y
pryor	y	n	n	y	n	y	y	n	y	y	n	y	y	y
bumpers	y	n	n	y	n	?	y	n	y	y	n	y	y	y
deconcini	y	y	n	y	y	y	y	y	y	y	n	y	y	y
heflin	y	y	n	n	y	y	y	y	n	y	y	y	y	n



