The Minimum Description Length Principle Applied to Feature Learning and Analogical Mapping

Mark Derthick
MCC
3500 West Balcones Center Drive
Austin, TX 78759
derthick@mcc.com

ACT-CYC-234-90

June, 1990

Abstract

This paper describes an algorithm for *orthogonal clustering.* That is, it finds *multiple* partitions of a domain. The Minimum Description Length (MDL) Principle is used to define a parameter-free evaluation function over all possible sets of partitions. In contrast, conventional clustering algorithms can only find a single partition of a set of data. While they can be applied iteratively to create hierarchies, these are limited to tree structures. Orthogonal clustering, on the other hand, cannot form hierarchies deeper than one layer. Ideally one would want an algorithm which does both. However there are important problems for which orthogonal clustering is desirable. In particular, orthogonal clusters correspond to feature vectors, which are widely used throughout cognitive science. Hopefully, orthogonal clusters will also be useful for finding analogies. A side effect which deserves more exploration is the induction of domain axioms in which the features are the predicates. The primary example used to demonstrate the orthogonal clustering algorithm, called MDL/OC, is finding the features {person, nationality, sex, generation} from the database of family relations used by Geoffrey Hinton [1986] to demonstrate feature discovery by a back-propagation network. Brief examples from the literature of clustering and analogical mapping are also given, to illustrate the generality of the technique.

Contents

1 Introduction **1**

2 Clustering **2**

2.1 Attributes versus Individuals 6
2.2 Problem Solving versus Knowledge Integration 6
2.3 Orthogonal versus Hierarchical Clusters 9
2.4 Supervised versus Unsupervised Learning 10
2.5 The New Term Problem 10
2.6 Noise 11
2.7 Continuous versus Discrete Features 11
2.8 Explicit versus Implicit Evaluation Function 11

3 Minimum Description Length Principle **12**

3.1 General Principles 12
3.2 Entropy 15
3.3 Notation 16
3.4 MDL Orthogonal Clustering 16
3.5 Search Algorithm 21
3.6 Family Relations Problem 23
3.7 Scaling 24
3.8 Prediction 25
3.9 Soybean Disease Problem 26
3.10 Country Trading Problem 27

4 Analogical Reasoning **30**

4.1 MDL/OC Considered Analogical 30
4.2 Direct MDL Mapping Algorithm 32

5 Future Work **36**

5.1 Knowledge Integration 36
5.2 Reasoning 38

6 Conclusion **38**

A Hinton's Algorithm **41**

B Interdependence-based Algorithm **46**
B.1 More Information Theory 47
B.2 Evaluation Function 47
B.3 Search Algorithms 50
B.3.1 Direct Approach 50
B.3.2 Successive Refinement 52

C Comparison of Evaluation Functions **55**

D Desired Features for Family Relations Problem **61**

1 Introduction

[[Say any self-supervised connectionist net can learn multiple independent features, with the same advantages and disadvantages as Hinton86]]

[[Say completion- and whole-tuple- algorithms for soybean problem performed similarly]]

This research is being carried out in the context of the CYC project, a ten year effort to build a program with common sense [Lenat and Guha, 1990]. Much of the effort is devoted to building a knowledge base of unprecedented size. In such a large KB there will inevitably be important concepts, relations, and assertions left out, even within areas that have been largely axiomatized. Inductive learning algorithms which can discover some of this missing information would be helpful. Conversely, having a large heterogeneous KB as a testbed is useful for exploring new learning algorithms.

The line of research reported here has steadfastly concentrated on finding an algorithm for assigning *features*[1] to individuals based on training examples consisting of tuples of those individuals. Different features can be thought of as representing *orthogonal clusterings*, where the number of possible values of each feature corresponds to the number of clusters in the corresponding clustering. Hopefully such an algorithm can be used directly to discover useful new concepts in the CYC KB, and can be used indirectly in analogical reasoning in CYC. Solving Hinton's [1986] family relations problem in a more pleasing manner has been the first important milestone. The algorithm which finally achieves this goal bears only a faint resemblance to the first attempt. Some of this history is described in appendices A-C.

Discovering feature-predicates and axioms is forming a more abstract and general theory of the domain than the theory consisting simply of the training tuples. The more powerful each axiom is, the fewer that are needed to describe the domain. Under the Minimum Description Length (MDL) paradigm, the size of the theory is quantified and serves as an evaluation of the theory.

The paper begins with a survey of the characteristics of clustering problems and algorithms in order to orient orthogonal clustering on the conceptual map. For concreteness, these characteristics are illustrated with respect

[1] Features are sometimes called "attributes" in the machine learning literature. The term "feature" connotes sets of properties that are largely independent, and directly useful for expressing domain knowledge.

to two problems, Hinton's family relations problem and Ryszard Michalski and R. L. Chilausky's [1980] soybean disease diagnosis problem. The MDL paradigm is explained, followed by its particular use in clustering with the MDL/OC algorithm. The application of MDL/OC to the family relations problem, soybean problem, and some real data from CYC are described, along with a related algorithm for finding analogical mappings. A fully satisfactory means of doing this has yet to be found, but some possible approaches are outlined, as well as outlines for discovering more abstract domain axioms.

2 Clustering

Many clustering algorithms have been developed, both in statistics and in machine learning. Duda and Hart [1973] give a good introduction. Only two previous approaches for finding orthogonal clusters have been explored, however. Hinton [1986] used back propagation to learn features in a domain of family relationships. He called his feature vectors "distributed representations," because the representation of an individual is structured, rather than atomic as pointers are. His algorithm is described in appendix A. Information theoretic algorithms akin to the one described in appendix B have also been developed [Lucassen, 1983, Becker and Hinton, 1989, Galland and Hinton, 1990].

The family relations training data consists of the 112 3-tuples representing true family relationships in the family trees shown in figure 1. Some examples are shown in figure 2. The output might include partitions for SEX, PERSON, NATIONALITY, GENERATION, and BRANCH OF FAMILY.[2] From the system's point of view, the representation of each person and each relation is atomic, with no inherent similarity measure. That the category "Italian" is useful, for instance, is only implicit in the correlations in the training tuples.

Formally, let S be the set of l training tuples. In general, the number of components, n, can be arbitrary. However a common special case, at least when learning from data in a frame based knowledge representation system, will have $n = 3$. In the family relations problem, the three components will often be referred to as PERSON1, RELATION, and PERSON2, although the algorithm treats all components identically. The domain from which the

[2] The figures in appendix D define these solutions.

```
Christopher  =   Penelope        Andrew = Christine
                     |                          |
             ---------------            -----------------
             |             |            |              |
Margaret = Arthur         Victoria = James         Jennifer = Charles
                                   |
                             --------------
                             |            |
                           Colin      Charlotte
```

```
             Roberto = Maria             Pierro = Francesca
                     |                          |
             ---------------            -----------------
             |             |            |              |
   Gina = Emilio             Lucia = Marco           Angela = Tomaso
                                   |
                             --------------
                             |            |
                           Alfonso     Sophia
```

Figure 1: The family trees from which the 112 training tuples are derived. "=" means "spouse," and lines indicate ancestor relationships.

```
(CHRISTOPHER WIFE PENELOPE)
(CHRISTOPHER SON ARTHUR)
(CHRISTOPHER DAUGHTER VICTORIA)
(ANDREW WIFE CHRISTINE)
(ANDREW SON JAMES)
(ANDREW DAUGHTER JENNIFER)
(ARTHUR WIFE MARGARET)
```

Figure 2: A few examples of the 112 training tuples for the family relations problem, in the syntax accepted by MDL/OC. For convenience, the three components are referred to as PERSON1, RELATION, and PERSON2.

training set is constructed is called I, for "individuals." The size of this set is called q. In the family relations problem the individuals include both the persons and the relations, and $q = 36$.

The goal is to learn a function, f, which maps individuals onto feature-vectors. In the family relations domain, a solution might have $d = 4$ features, having the following values:

Feature	Arity	Values
SEX	2	{Male Female}
PERSON	2	{Person Relation}
NATIONALITY	2	{English Italian}
GENERATION	3	{1st-Gen 2nd-Gen 3rd-Gen}
BRANCH OF FAMILY	3	{Central Intermediate Outside}

The arity of feature i is c_i. With the above assignment, $f(Penelope) = < Female, Person, English, 1st - Gen, Central >$. The names of the features and values are chosen after the fact by the experimenter to simplify explanation. The algorithm only knows that Penelope has value 0 for feature 2, for example.

Features represent c_i-ary partitions of the domain. Each element of a partition, such as Male, is equivalent to a one-place predicate, so a feature can be represented in CYC as a set of mutually disjoint collections that cover the domain.

This says nothing about what makes a *good* f. Section 2.2 touches on

```
(DIAPORTHE-STEM-CANKER DATE-OCTOBER PLANT-STAND-NORMAL PRECIP-GT-NORM
        TEMP-NORM HAIL-YES CROP-HIST-SAME-LST-YR
        AREA-DAMAGED-LOW-AREAS SEVERITY-POT-SEVERE SEED-TMT-NONE
        GERMINATION-90-100% PLANT-GROWTH-ABNORM LEAVES-ABNORM
        LEAFSPOTS-HALO-ABSENT LEAFSPOTS-MARG-DNA LEAFSPOT-SIZE-DNA
        LEAF-SHREAD-ABSENT LEAF-MALF-ABSENT LEAF-MILD-ABSENT
        STEM-ABNORM LODGING-NO STEM-CANKERS-ABOVE-SEC-NDE
        CANKER-LESION-BROWN FRUITING-BODIES-PRESENT
        DECAY-FIRM-AND-DRY MYCELIUM-ABSENT INT-DISCOLOR-NONE
        SCLEROTIA-ABSENT FRUIT-PODS-NORM SPOTS-DNA SEED-NORM
        MOLD-GROWTH-ABSENT SEED-DISCOLOR-ABSENT SEED-SIZE-NORM
        SHRIVELING-ABSENT ROOTS-NORM)
```

Figure 3: One example of the 290 training tuples for the soybean disease diagnosis problem, in the syntax accepted by MDL/OC. The first component is the disease, and the rest are symptoms.

this, but specifics are deferred until section 3. This section primarily contrasts alternative ways of stating the problem, and alternative frameworks for finding good f's.

The machine learning database maintained at UC Irvine contains training data for several domains. One of these, first used by Michalski and Chilausky [1980] and subsequently by many others [Tan and Eshelman, 1988, Fisher and Schlimmer, 1988], contains attributes of diseased soybean plants together with their diagnoses. Each plant has exactly one of 15 diseases. An example training tuple is shown in figure 3. The desired output is the correct diagnosis of a plant's disease given its attribute values. Both the disease and the symptoms are represented as attribute-values. It is in fact a very common clustering task to learn to predict a given attribute given others. An alternative task is to learn to predict *any* attribute given some others. MDL/OC can be thought of as attacking the problem of predicting *all* attributes simultaneously.

2.1 Attributes versus Individuals

A major difference between the family relations problem and the soybean disease problem is how individuals are represented in the input. In the soybean problem, individuals are represented by their attribute values. When new individuals are encountered in the test set, generalization can be based on the similarity of their attribute values to those of individuals from the training set. For instance, the majority of individual plants with DIAPORTHE-STEM-CANKER might share the value AREA-DAMAGED-LOW-AREAS. But in the family relations problem, individuals are represented atomically. (Compare with the alternative representation in figure 4 where a training tuple represents a single individual.) Generalization can only be based on the context in which the novel individual occurs.

The orthogonal clustering algorithm is sufficiently flexible that it can be applied to the supervised soybean problem, in spite of this difference in representation. The results of this are described in detail in section 3.8. This is done by treating each value for each attribute as a distinct individual. Each disease is also treated as an individual. After feature vector assignments are learned for each attribute-value and each disease, the inference rules can be used to predict the diagnosis for novel inputs. Going the other way, it is difficult to see how traditional clustering algorithms can be applied to the family trees problem without losing the relationship between, for instance, Penelope as PERSON1 and Penelope as PERSON2. In spite of the fact that the role Penelope plays in an assertion affects the predictions that can be made, in support of the goal of equating feature values with concepts in a taxonomic knowledge base there must be a context-independent set of concepts a given individual instantiates. Using the representation in figure 4 does not help, because Penelope will still appear in multiple columns. Further, attributes cannot have multiple values, as would be required by the natural representation of Colin's aunt attribute.

2.2 Problem Solving versus Knowledge Integration

Most clustering algorithms are oriented towards performance on some task. The learned parameters serve to predict some attribute of new examples, but this procedure often takes place in a black box as far as the end user is concerned. Michie and Al-Attar [1990] argue that algorithms which learn

Individual	husband	wife	brother	sister	son	daughter	aunt
Penelope	Christopher	-	-	-	Arthur	Victoria	-
Colin	-	-	-	Charlotte	-	-	{Margaret Jennifer}

Figure 4: An attribute/value representation of two individuals from the family relations domain. (Only 7 of 12 attributes are shown.)

decision-trees also provide insight into the domain structure, because the procedure of narrowing down the final cluster by sequential tests on the attributes is a familiar form of reasoning. For instance, the tree in figure 5 can be used to associate a newly observed individual with previous individuals that are similar. Beginning at the top, it is first grouped with the partition of the domain with which it shares a value for the Person/Relation feature. Within this partition, it is then associated with the subpartition with the same value for the Sex feature. At the bottom of the tree, it will be grouped with individuals sharing values for all attributes tested on the path from the root. If the tree is organized so the most important attributes are at the top of the tree, and irrelevant attributes are not tested, this will provide a statistically useful sample of similar individuals from which to guess the value of unknown attributes.

That this inductive inference procedure can be applied with understanding by a person is undeniable when each test is intuitively meaningful and there aren't too many of them on the route to a conclusion. In such cases, each route can be thought of as a production rule [Quinlan, 1987], with all the tests on the left hand side, and the predicted attribute value on the right hand side. Since all these terms are expressed in the natural language of the domain, it ought to be possible to compare them to hand-entered rules in an expert system. Rules expressing essentially the same knowledge can be combined, and new rules can be added.

It seems a forlorn hope that this process can be done without a human intermediary who can see the relationships between the induced and hand-entered rules and properly integrate the new knowledge. In large domains, it is important to restrict the complexity of the induced regularities sufficiently so that the human interpreter has a chance of gaining an intuitive

```
                     /\
                   /    \
                 /        \
            Person          Relation
             /  \             /   \
           /      \         /       \
     Female        Male   FemaleRel MaleRel
     /|\            /| \    / | \     / | \
   /  |  \        /  |   \  S O  Y  S' O'  Y'
  1    2   3     1'  2'   3'
 /\   /\   /\   / \  / \   / \
E  I E' I'E" I"E"'I"'E""I""E""'I""'
```

Figure 5: Possible hierarchical clustering of family relations domain. S=Same Generation; O=Older; Y=Younger; 1=1st Generation; 2=2nd Generation; 3=3rd Generation; E=English; I=Italian. Primes indicate rediscovery of the "same" distinction over different subsets of the training set.

understanding of them. I believe intuitively good features should have two characteristics over and above facilitating predictions about unknown components of a tuple: Prediction should be simple, and the features should not be redundant. Therefore an important characteristic of MDL/OC is that it searches for a very constrained kind of regularity: correlations among values of a *single* feature across the training tuples. It gets its power from discovering new features which make these underlying regularities apparent. After learning, the human interpreter must puzzle over the feature assignments and resulting correlations before integrating the new knowledge into CYC. A side effect of only finding a certain kind of regularity is that the discovered domain theory will usually be incomplete, and therefore this kind of algorithm will do poorly as a black-box problem solver. For this task, traditional algorithms that only optimize predictivity are better.

```
/ | \
     /\              / \              / \                 /   |   \
   /    \          /     \          /     \             /     |     \
Female  Male   English Italian   Person Relation  1st Gen  2nd Gen  3rd Gen
```

Figure 6: Possible orthogonal clustering of family relations domain

2.3 Orthogonal versus Hierarchical Clusters

In a hierarchical classification such as figure 5, the data are sequentially partitioned into smaller and smaller ever more specialized groups. In the main, this seems to mirror human organization of world knowledge. However it is restrictive that a tree-structured hierarchy can only categorize one way. One could study the same work in a class on Russian literature, psychological novels, or political philosophy in the time of Napoleon. The family relations domain was chosen for its extreme orthogonality. Except for the dependence of NATIONALITY (and to a lesser extent the other features as well) on PERSON, the serial dependence inherent in a hierarchical organization is artifactual. This property of attributes may not be a common one, but it is a valuable one. To the extent that features contribute independently to inference, a domain obeys the principle of superposition, the hallmark of linear systems in engineering disciplines. The convenience afforded by assumptions of independence is embraced almost universally in Bayesian inference, and often even in the very systems that learn decision trees. It is surprising that the virtues of an orthogonal feature set are recognized, yet there is almost no attempt to discover orthogonal clusters. Perhaps it is because there is little need to combine the effects of arbitrary constraints on arbitrary combinations of attributes.

Aside from advantage of simple evidence combination rules, orthogonal clustering algorithms do not suffer from a combinatorial reduction in the amount of data available as the tree grows. Finally, finding regularities that apply across the whole domain may reduce the need for explicit analogical reasoning (see section 4).

2.4 Supervised versus Unsupervised Learning

One important distinction to be made among inductive learning algorithms is whether they are supervised or unsupervised. Either type can be applied to the soybean problem. In the supervised case, the algorithm is provided with the correct diagnosis for each training case. There is the possibility for "cheating" by memorizing the diagnosis for each case rather than learning general predictive relationships between attribute values and diagnoses. So the results of these algorithms are usually evaluated on a disjoint set of examples from the training set, called the test set.

In the unsupervised case, the algorithm is given only the attribute values and it must decide what diseases there are as well as how to map from attribute values to diagnoses. There is no guarantee that the disease classes found by the algorithm will have any relation to the 15 defined by human experts. So for this problem a supervised algorithm would probably be more useful. More generally, whenever the desired output data for a problem is available, supervised learning is usually preferred.

Both supervised and unsupervised algorithms can be applied to the family relations problem, too, although the former is unnatural. In order to do supervised learning, there must be a desired output for each training instance. Since the ultimate goal is a mapping from individuals to feature vectors, the desired output can be the desired feature vector for each component of the input tuple. The algorithm described in this paper is unsupervised, because the goal is to learn new concepts in which to express the domain theory, rather than just to extend a theory using known concepts. In problems where the primitives are individuals, as opposed to attributes, forming new concepts is necessary, because the known ones are not useful for expressing an abstract domain theory.

2.5 The New Term Problem

Except for the problem of ignoring identity of individuals across components of the training set, there is a way to describe the family relations problem in current ML terminology. The so-called "new term problem," or "constructive induction" [Michalski, 1983], is to redescribe the input by more useful attributes and/or values. It is presumably easier to learn which tuples occur in the training set if the tuples are already expanded out into, for example, {old central male Italian, same-generation opposite-sex opposite-branch same-country, old central female Italian} rather than {Pierro, wife, Lucia}. Each of these new terms is just a disjunction of values of some existing attribute. None of the current algorithms in the constructive induction literature can solve the family relations problem, however.

2.6 Noise

In training from examples, noise consists of examples which are not in fact instances of the concept they are purported to be.[3] For some (usually highly-biased and fast) algorithms, noise is intolerable. For orthogonal clustering, and many others, performance gradually degrades with increasing noise. There are even some algorithms that must have noise for best performance. In the results reported for the family relations problem, there was no noise. The data taken from the CYC KB are presumably noisy.

2.7 Continuous versus Discrete Features

Some features, like sex, are inherently discrete, while others, like age, are inherently continuous. MDL/OC can only learn discrete features (but see appendix B.3.1). If age is a useful feature for some domain, the algorithm will discretize the range into useful subsets, but it cannot take into account the ordering of the subsets.

2.8 Explicit versus Implicit Evaluation Function

In MDL/OC there is a strong distinction between the declarative "evaluation function" which rates proposed solutions and the procedure used to find candidate solutions. An evaluation function is one of several types of "testers," which also could be comparison predicates, optimality predicates, or satisficing predicates. In contrast, hierarchical clustering algorithms usually have no declarative specification of what is an optimal (or satisfactory) solution for a given training set. Rather they have sequential algorithms that find an initial partition of the individuals, each of which is in turn partitioned. After an

[3] Pedagogically, consider negative instances to be instances of the complement concept, and consider the unsupervised case to be learning a single concept of which all the training examples are instances.

initial tree is built, it is often modified to eliminate spurrious distinctions or combine common substructure. The combined effect of these separate steps are obviously hard to characterize. Solutions are tested, of course, but usually by completion performance on a test set. But completion performance is not a declarative specification from which a procedural implementation can be derived. For instance, it is not the sort of evaluation function one can hill-climb on, because by definition the testing set is not to be used during training.

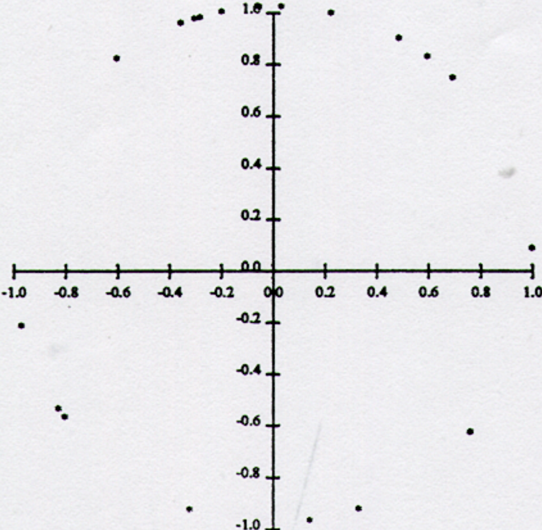
AUTO-CLASS [Cheeseman *et al.*, 1988] is a clustering algorithm using an attribute/value representation which does have a declarative evaluation function which plays the right kind of causal role in the search algorithm. AUTO-CLASS is based on Bayesian inference, which has a close relation to MDL. AUTO-CLASS also shares with orthogonal clustering the characteristic that it is not hierarchical. Cheeseman *et. al.* point out that there are techniques for building a taxonomy from the leaf concepts after learning is completed. The information theoretic algorithms mentioned above [Lucassen, 1983, Becker and Hinton, 1989, Galland and Hinton, 1990] also use a declarative evaluation function.

3 Minimum Description Length Principle

3.1 General Principles

MDL is a very powerful and general approach which can be applied to any inductive learning task. It appeals to Occam's razor—the intuition that the simplest theory which explains the data is the best one. The simplicity of the theory is judged by its length in some language chosen subjectively by the experimenter. Its ability to explain the data is measured by the number of bits required to describe the data given the theory. A complete theory would require no bits for this. The example data set in figure 7 illustrates the compression achievable with a good theory. In general, encoding points in a plane requires two numbers each. However if the theory includes the constraint that $x^2 + y^2 = 1$, then each point only requires one number, to encode the angle.

MDL is also the goal of coding theory, in which the problem is to communicate a given message through a given communication channel in the least


1.0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
-0.8
-1.0
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1.0

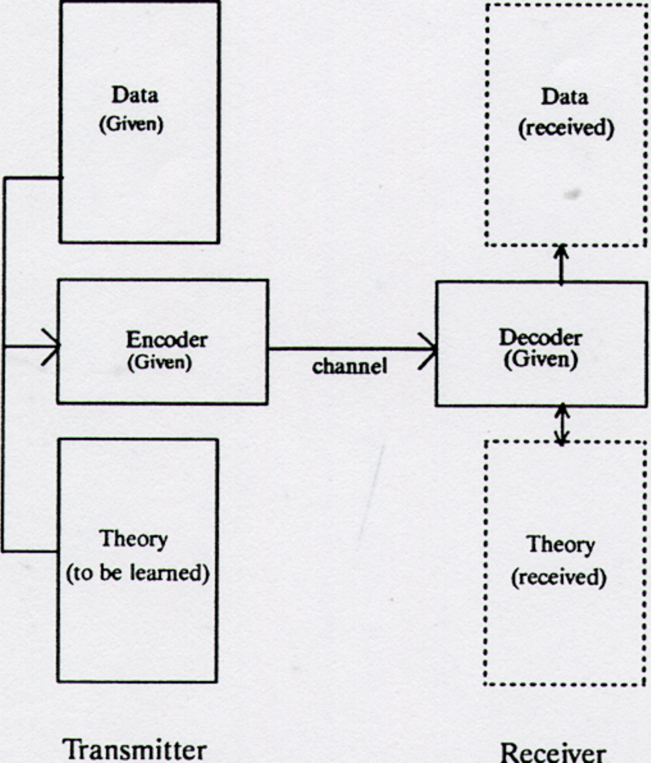

Data
(Given)
Data
(received)
Encoder
(Given)
channel
Decoder
(Given)
Theory
(to be learned)
Theory
(received)
Transmitter
Receiver

Figure 7: A set of points with an obvious regularity, allowing significant compression.

Figure 8: The MDL paradigm is conceptualized as a communication task, in which the sender knows some data he wants to transmit to the receiver. They are given ahead of time common languages for expressing the data and domain theory, and a common encoding/decoding scheme. The transmitter's goal is to learn a theory, which he must transmit to the receiver, that will minimize the total number of bits that must be sent across the communication channel yet still allow the receiver to recover the data.

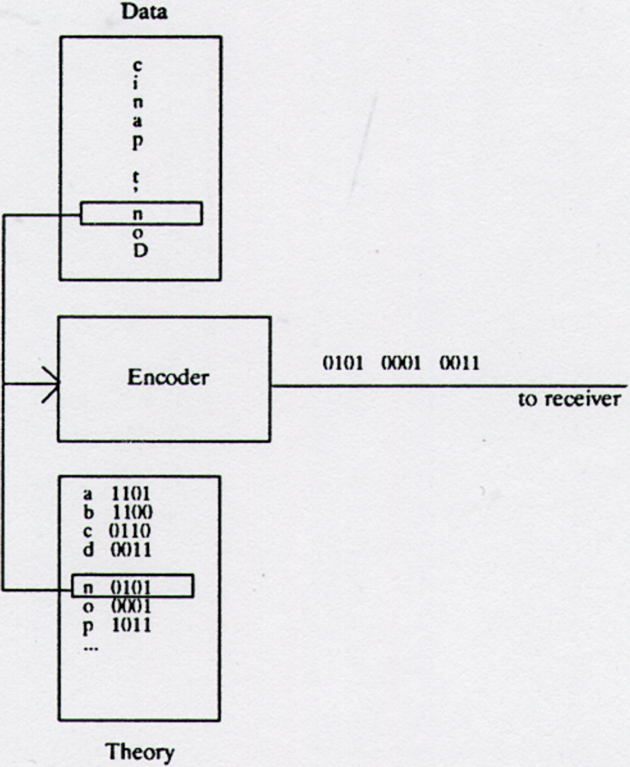
time or with the least power. Raw source data is first encoded, then transmitted, and finally decoded (see figure 8). The more sophisticated the theory of the source domain, the greater the compression of the data that can be achieved. But the theory must also be transmitted, so there is a trade-off. If there is lots of data, it is worth spending more time on the "overhead" of transmitting the theory.

Figure 9 gives a concrete example of an encoding scheme in which each letter of an English text is transmitted individually. The "theory" in this case consists simply of a lookup-table of code-words that go with each character. This would be a good scheme if each letter has a probability which is time- and history- invariant. In this case there is a simple algorithm for finding an optimal code, in which the most frequent characters have the shortest code. This technique is called "Huffman coding."

For a message shorter than the size of the ASCII character set, this coding scheme will be inferior to just transmitting the original ASCII. And for sufficiently long messages it will be worth coding common pairs of characters, common words, common phrases, common topics, common forms of argument, and possibly all the way up to complete psychological theories of human cognition.

A very elegant encoding scheme is using a Universal Turing Machine. Then the theory to be learned is a Turing Machine program. In this case the

Figure 9: Illustrative example of an encoder, data set, and theory, and the resulting transmitted message.


Data
c
i
n
a
p
t
,
n
o
D
Encoder
0101 0001 0011
to receiver
a 1101
b 1100
c 0110
d 0011
n 0101
o 0001
p 1011
...
Theory

length of the theory description plus data description for the optimal theory is the data's algorithmic complexity [Chaitin, 1977], which is sometimes also called its Kolmogorov complexity. This is not a computable function, and the space of Turing Machine programs is not a very good one in which to do heuristic search. So for inductive learning, an encoder better tailored to the specific task is chosen by the experimenter. Of course there is no guarantee that an intuitively constructed language will have the expressive power to formulate the best theory, nor is there even a guarantee that if the language is powerful enough to express the intuitively best theory, it will have the shortest expression *in that language*. The justification of such an approach must rest on a subjective evaluation of the language.

In machine learning, subjectivity arising from the language for formulating theories is considered a bias. As described by Gao and Li [1989], Rissanen [1978] has provided an elegant analysis of the bias in MDL learning systems in terms of Bayesian inference. By Bayes' rule

$$\Pr(T|O) = \frac{\Pr(O|T)\Pr(T)}{\Pr(O)}$$

Using maximum likelihood inference, the best theory (T) is the one whose posterior probability given the observations (O) is highest. The prior probability of the observations is a constant with respect to this choice, so only the product $\Pr(O|T)\Pr(T)$ is of concern. If these probabilities are defined in terms of the data-term and theory lengths such that the lengths are inversely proportional to the logarithms of these probabilities, then the MDL principle can be reduced to the maximum-likelihood principle.

Intuitively, in the family relations data, nationality is a useful feature because there is a strong sub-regularity in the tuples, such that there is a lot of uncertainty about the nationality of any given *component* of any given tuple, but there is very little uncertainty about the patterns of feature-values *across* tuples. The latter means it takes little information to code what class of tuple we are talking about (1 bit, for English vs. Italian), and the latter means it provides a lot of information about which individuals fill the tuple (2 bits, one for PERSON1 and one for PERSON2). So a theory in which individuals are mapped to feature vectors, feature tuples are encoded and transmitted, and then decoded again by the receiver, makes intuitive sense (see figure 10).

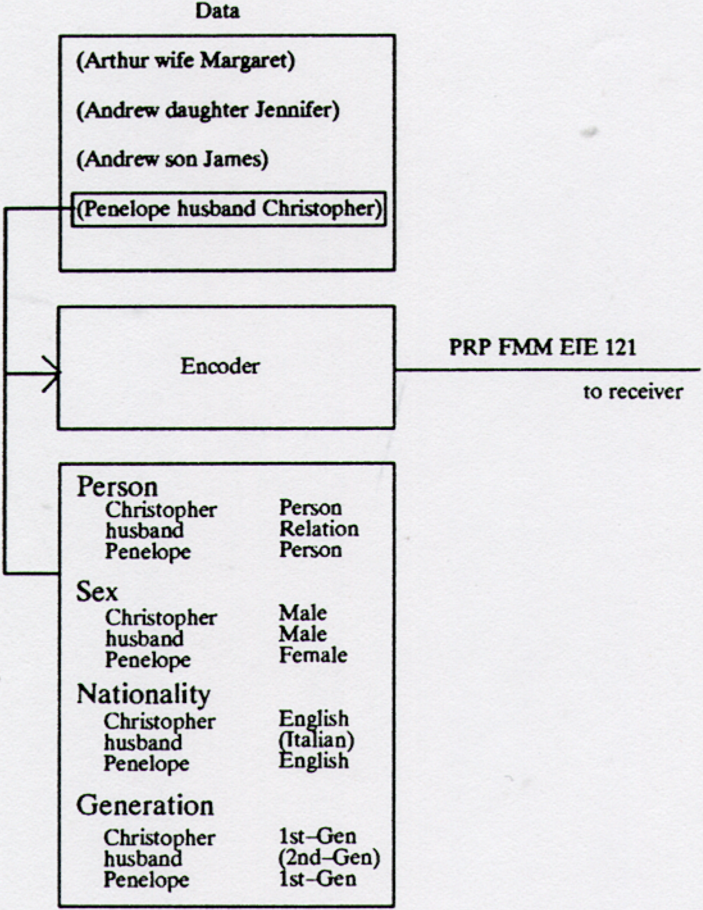

Data
(Arthur wife Margaret)
(Andrew daughter Jennifer)
(Andrew son James)
(Penelope husband Christopher)
Encoder
PRP FMM EIE 121
to receiver
Person
Christopher Person
husband Relation
Penelope Person
Sex
Christopher Male
husband Male
Penelope Female
Nationality
Christopher English
husband (Italian)
Penelope English
Generation
Christopher 1st–Gen
husband (2nd–Gen)
Penelope 1st–Gen

Figure 10: The transmitter sends an input tuple by sequentially mapping it onto feature-tuples, one for each feature, of which there are four here. Each input 3-tuple can thus be transmitted as a sequence of four patterns of feature values. The advantage is that, if good features can be found, the number of alternative patterns is much smaller than the number of individuals, so shorter codes can be used. At the destination, the process is just the reverse. The code-words representing feature-value patterns are decoded into the features, and vectors of features are decoded into individuals. In case multiple individuals have the same feature vector, the code-word is augmented to disambiguate. The feature value names in this figure are attached by the experimenter as an aid to understanding. When applied to RELATION, "2nd Gen" doesn't really mean second generation, but rather same generation. The name is in parentheses to indicate that while the value is formally "2nd Gen," the label is not helpful in this case. Similarly "Italian" is not a helpful label when applied to a RELATION.

3.2 Entropy

To quantify the length of a coded message, information theorists use the concept of entropy. The entropy of a random variable, $\mathcal{S}$, measures how much uncertainty there is about its value. It can be thought of as "negative information" in that it represents how much more would have to be known to pin down the value exactly. Entropy is defined to be $H(\mathcal{S}) = -\sum_{\alpha \in \mathrm{domain}(\mathcal{S})} \Pr(\alpha) \log \Pr(\alpha)$. The base of the logarithm determines the units in which the entropy is measured. Bits are commonly used in computer science, so all logarithms in this paper are base two. It is a theorem that the shortest possible average length for any code for a sample sequence of values of any random variable is the entropy of that variable. Further, given a sufficiently long sequence of source values to encode, there exists a code which approaches this limit arbitrarily closely.

If there are two random variables, the *joint entropy*, $H(\mathcal{U}, \mathcal{S})$, measures the uncertainty about the cross-product space, and the conditional entropy measures the residual uncertainty about one when the other is known. Formally, $H(\mathcal{U}|\mathcal{S}) = H(\mathcal{U}, \mathcal{S}) - H(\mathcal{S})$. If the conditional entropy $H(\mathcal{U}|\mathcal{S})$ is lower than the unconditional entropy $H(\mathcal{U})$ then $\mathcal{S}$ must bear information about $\mathcal{U}$. Textbooks such as Chernoff and Moses [1959] elaborate on these concepts.

3.3 Notation

Sets are denoted by capital roman letters, vectors have arrows over them, and random variables are in caligraphic script. J_+ stands for the set of positive integers, and $J_{1,c}$ stands for the set of integers between 1 and c.

In the orthogonal clustering task, a set I of individuals is given, which is denoted $I_k, k = 1, q$. Out of these are composed a set S of n-tuples, $S_l, l = 1, m$. Taking these tuples as training examples, the task is to assign feature vectors $\vec{x}$ to individuals. Feature vector components are denoted $x_i, i = 1, d$. The arity of each feature is $c_i, i = 1, d$, so each feature can take on integer values between 1 and c_i. The function $f : I \to \prod_{i=1}^{d} J_{1,c_i}$ maps individuals onto their feature vector. $f_i : I \to J_{1,c_i}$ picks out the value of a single feature. Sometimes it is convenient to find the value of a single feature for each component of a tuple, so f is overloaded by allowing it to apply to a tuple of individuals and return a vector of results: $f_i : S \to \prod_{j=1}^{n} J_{1,c_j}$. Such a vector is represented by the variable $\vec{y}$, with components $y_j, j = 1, n$.

The relative frequency with which a given tuple occurs in the training set is $\Pr(\mathcal{S} = \vec{t}), \vec{t} \in I^n$. Although the training set is regarded as a sample from an unknown distribution, the notation does not bother to distinguish sample probabilities from true probabilities. To get at parts of input tuples, feature vectors, and parts of feature vectors, $\Pr(\mathcal{S}_j = I_k)$, $\Pr(f_i(\mathcal{S}) = \vec{y})$, and $\Pr(f_i(\mathcal{S}_j) = y_j)$ are also useful. $\mathcal{T}$ is a random variable ranging over individuals with probabilities derived from the training set: $\forall i \in I \Pr(\mathcal{T} = i) = \frac{1}{n} \sum_{j=1}^{n} \Pr(\mathcal{S}_j = i)$. Probabilities will always be summed over all possible states of a random variable, so the explicit reference to the variable standing for the state can often be dropped. Now the entropy of the probability distribution induced by mapping the training tuples onto feature tuples can be defined as $H(f_i(\mathcal{S})) = \sum \Pr(f_i(\mathcal{S})) \cdot \log \Pr(f_i(\mathcal{S}))$. Similarly, the joint entropy induced by mapping the training tuples onto feature-vector tuples is $H(f(\mathcal{S})) = \sum \Pr(f(\mathcal{S})) \cdot \log \Pr(f(\mathcal{S}))$.

3.4 MDL Orthogonal Clustering

The encoding scheme diagrammed in figure 10 seems to be transmitting more information than necessary, in that the order doesn't seem to be important at all. If there were a better way to transmit the set as a whole, unordered, it would seem more natural. I tried a scheme where the features were transmitted, as well as which patterns of feature-values occurred. Then the receiver reconstructed all possible tuples that fit these patterns. The expression for the total length was expensive to compute, so I never tried to learn with it. Further, it produced a non-intuitive ranking of possible solutions. The basic problem was that it made far too many predictions. It's not every middle-generation, left-side-of-the-family, English woman that marries every middle-generation, left-side-of-the-family, English man, after all. I concluded that coding every training example separately wasn't as wasteful as it first seemed.

A second MDL scheme I rejected was optimizing the answering of questions about any one component of the tuple given the other two. The training set of 112 tuples was treated as 336 tuples in which two components were inputs and one was output. The rankings for this method were more intuitive, but when free to learn its own features it preferred some bizarre ones to these (see appendix C and figure 33). By being able to count on having two components always known, it did not have to model the interdependency of each feature completely, and did some non-intuitive overloading. For instance its version of branch of the family had all of Penelope's descendents plus Charles grouped together. Sticking Charles on is useful because it differentiates him from Jennifer, as far as being a descendent of Christine, while there is presumably other information available to distinguish him from Penelope's real descendents.

For the coding scheme previously diagrammed in figure 10, both the rankings and the free choices are intuitive. The expression for total length under this scheme is therefore given in detail below. This total includes both the background theory that defines the features and lists the feature-tuples that occur, and the data description itself. The data description conceptually has two parts: to transmit a single training tuple, the set of feature-tuples it maps to, the $f(\mathcal{S})$, must be transmitted. Then, if $f(\mathcal{S}_j)$ isn't one-to-one, disambiguating information must be transmitted to pick out the correct inverse. In general, the more sophisticated the theory, the longer the feature-tuples codes, but the shorter the disambiguating information. Two extremes are worth examining. The universal feature has arity one, so requires zero feature-tuple information. The disambiguating information must identify every component of every training tuple from scratch. At the other extreme is the identity feature, which has a different value for every individual. In this case no disambiguating information is necessary, but the theory is enormous

Theory	
Number of individuals	$\log *(q)$
Number of training tuples	$\log *(l)$
Arity of training tuples	$\log *(n)$
Number of features	$\log *(d)$
Feature Arities	$\sum_{i=1}^{d} \log *(c_i)$
Feature Assignments	$q\sum_{i=1}^{d} \log(c_i)$
Code lengths and Codes for Feature Tuples	$\sum_{i=1}^{d} c_i^n(\log *H(f_i(\mathcal{S})) + H(f_i(\mathcal{S})))$
Code lengths and Codes for Disambiguation	$-\sum_{k\in individuals} \log * \Pr(\mathcal{T} = k \mid f(\mathcal{T}) = f(k))$
Data	
Feature Tuple Codes	$l\sum_{i=1}^{d} H(f_i(\mathcal{S}))$
Disambiguation Codes	$nlH(\mathcal{T}\|f(\mathcal{T}))$

Figure 11: Breakdown of the contributions to the total description length of the training set using the encoding scheme pictured in figure 10.

since codes must be defined for the 112 patterns that actually occur out of a possible space of over a million (q^n).

In English, what is transmitted is the number of features, their arities, the assignment of individuals to feature vectors, the codes for each pattern of feature-tuples, and the extra disambiguation codes which are transmitted for each component of the training tuple for which that individual's feature vector is ambiguous. Figure 11 summarizes the contributions to the total description length. Aspects of the expression in the figure are explained below:

Unbounded Whole Numbers To encode the number a when the receiver does not know its range, not only must $\log a$ bits be transmitted, but also the number of bits to be transmitted, $\log\log a$. But the number of bits in this must also be transmitted, until the process grounds out in a pre-established number of bits, or in a unary representation. In the figure, these lengths are represented as $\log *a \equiv \sum_{k=1}^{b} \log^k a$, where the summation includes all positive terms. But this level of precision seems unwarranted, so they are approximated as just $\log a$.

Feature Arities The feature arities, c_i, figure prominently in the length expression, but this is a very bad measure in which to do hill-climbing, because it is so discrete. If the system had almost discovered sex, but assigned both Charles and Gina a third value (which has no correlate in Western gender models), it would be penalized for having a 3-ary feature even though it is very close to having a binary one. The hill-climbing search proceedure used below considers moves in which a single individual has its feature vector changed. But even if it considers making Charles male, it will not see any reward as far as reducing the arity. So for pragmatic reasons, the log of the arity of a feature is approximated by the entropy of its probability distribution over all occurrences of all individuals in the training set, $\log c_i \approx H(f_i(\mathcal{T}))$. When the feature partitions the individuals so that each value occurs equally often in the training set, this approximation is exact. As the partition becomes more uneven, the approximation varies smoothly down towards the next lower arity.

Feature-tuple Coding The length expression for transmitting the feature-tuple code is also an approximation in the interest of smoothness and computational convenience. The information required is a lookup table so that any pattern code can be decoded into a pattern. To make the data term most efficient, we should use a Huffman code. Then in the ideal case, the pattern code length for any pattern is minus its log probability. Rare codes have long patterns, but since they are rare they don't contribute much to the data term length, because $\lim_{p\to 0} p \log p = 0$. But when transmitting the code itself, the average over the pattern probabilities does not come into play, so the table length would be $-\sum_{z \in patterns} \log \Pr(z)$ which diverges as any of the probabilities approach zero. So this is a bad scheme to use when probabilities get very small. An alternative which avoids the problem of not transmitting the codes directly is described in appendix C. It also avoids the overhead of saying of unused feature tuples that they are unused, which prevents the evaluation function from blowing up as c_i or n increases. But it is less smooth and hence less conducive to hill-climbing. In any case, the length expression in the table uses the entropy of the feature as the average pattern code length, which corresponds to averaging the code lengths with a weight of the observed probabilities in the data rather than with the uniform probability $\frac{1}{c_i^n}$.

Disambiguation Coding The same trick is used for transmitting the disambiguation code. The average code length is approximated as the conditional entropy of the individual given its feature vector. The total is therefore $qH(\mathcal{T}|f(\mathcal{T}))$.

Number of Features The value of d should be the number of non-trivial partitions, but this has no smooth approximation that seems very meaningful. It is expected to be very small, so like the residual logarithms of logarithms, it is dropped.

Feature Tuples For the data term, the simplifying assumption is made that a perfect code can be found for the patterns. Such a code would have an average length of $H(f_i(\mathcal{S}))$, so the data term for encoding the feature value patterns is $l \sum_i H(f_i(\mathcal{S}))$.

Disambiguation Information When the d patterns are combined to form n feature vectors, any of the n could be ambiguous. Again assuming a perfect code, disambiguating the individuals will take $nlH(\mathcal{T}|f(\mathcal{T}))$.[4] Since f is a deterministic function, $H(\mathcal{T}, f(\mathcal{T})) = H(\mathcal{T})$, so $H(\mathcal{T}|f(\mathcal{T})) = H(\mathcal{T}) - H(f(\mathcal{T}))$.

The expression to minimize, including both theory and data terms, and all the approximations, is

$$E(f) = \sum_{i=1}^{d} \Big[(q+1)H(f_i(\mathcal{T})) + H(f_i(\mathcal{S}))(e^{nH(f_i(\mathcal{T}))} + l)\Big] + (nl+q)\cdot(H(\mathcal{T}) - H(f(\mathcal{T}))) + \log qln$$

The constant terms, $H(\mathcal{T})$ and $\log qln$, can be ignored by the optimization algorithm. The values reported in this paper include the former contribution, but not the latter.

[4] Instead of one disambiguation code used for all components, n component-specific codes could be used. This would increase this contribution to the theory size by nearly a factor of n, but if the component-specific distributions were significantly different from the averaged distribution, it might be worth it. This possibility is examined in appendix C.

Features	Theory	Feature Tuples	Disambiguation	Total
(Person Sex Nationality Generation)	284	625	339	1248
(Person Sex Nationality Generation Branch)	335	922	78	1334
(Person Generation)	251	296	899	1446
(Person Sex)	201	218	1066	1485
(Person Nationality)	201	113	1179	1492
(Person Branch)	224	296	1036	1557
(Person)	185	0	1402	1587
()	183	0	1711	1894
(RANDOM)	208	335	1375	1919

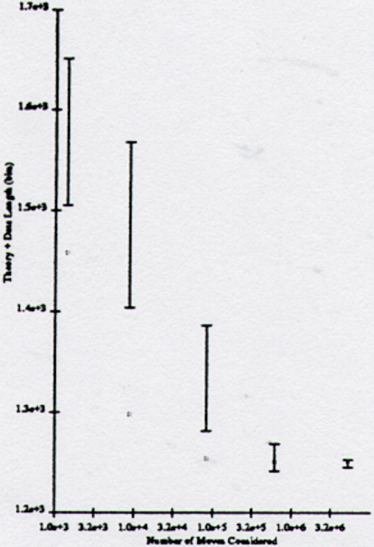
Table 1: The value of the evaluation function for several sets of features. The units are bits.

Table 1 lists the value of this function for several sets of features. Generally, the rankings accord with intuition. Appendix C describes variations on the evaluation function and the resultant differences in feature preferences.

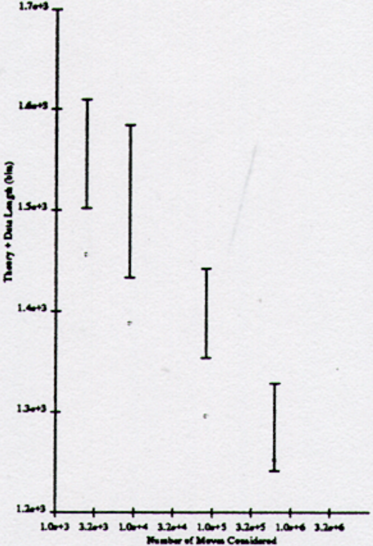
3.5 Search Algorithm

At this point the orthogonal clustering problem has been cast as a well defined optimization problem: minimize $E(f)$ over all possible sets of partitions of I, using sample probabilities from S. It is natural to think of the problem as one of placing the individuals in a discrete space of d dimensions. There is a one-to-one correspondence between coordinate assignments in this space and sets of partitions (see figure 12). The feature arity is limited by the number of possible coordinates along each dimension. Initially the individuals are assigned distinct random coordinates. Neighboring feature sets in the search space are those for which only a single coordinate of a single individual differs. Figure 13 shows the results of hill climbing (the leftmost data point in each graph) and simulated annealing (the other data points) for the family relations problem. Local optima are a serious problem for straight hill-climbing, as none of the solutions approach the presumed global optimum of sex, person, nationality, and generation.

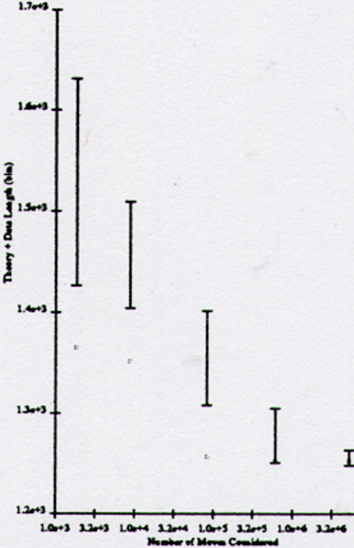
Using simulated annealing, the results are better. This is a generalization of hill-climbing in which the decision of whether to accept a proposed move is



Scaling for Five Binary Features



Scaling for Five Ternary Features



Scaling for Four Ternary Features

FEATURE	PARTITION
sex	((Penelope Lucia) (Charles Emilio))
nationality	((Penelope Charles) (Lucia Emilio))

	sex	nationality
Charles	1	0
Emilio	1	1
Penelope	0	0
Lucia	0	1

Figure 12: Any set of d partitions, each of arity c_i, can be represented by assigning to individuals a location (vector) in d-space, where there are c_i discrete coordinates along dimension i. Here two binary features are shown both as partitions and as assignments of feature vectors to individuals. The hill-climbing algorithm considers moves in which one feature-vector component of one individual is changed. For example, the nationality component of Lucia's feature vector might be changed to 0.

Figure 13: The theory length is plotted against the number of moves considered during the search for three limits on the feature sizes. In each case, the leftmost data point is zero-temperature annealing, which is equivalent to hill climbing. The other data points anneal from an initial temperature of 500.0 and gradually decrease until the probability of accepting any move fell below 0.001. This happens around a temperature of 1.0 for this problem. Each time a move is considered, the temperature is multiplied by a constant. Successive data points were derived by setting this constant to .999, .9999, .99999, and .999999. The slowest rate represents about four hours per trial on a Symbolics 3630. On the left, the search space includes five binary features; the middle graph comes from searching for five ternary features; and on the right it includes four ternary features. The error bars extend one standard deviation above and below the mean. The asterisks indicate the best solution obtained (over 20 runs in each case).

non-deterministic. The greater the improvement in the evaluation function (which is called the *energy function* in simulated annealing), the greater the chances of accepting a move. But even for moves that worsen the evaluation function, there is some chance of being accepted. Numerically,

$$\Pr(move) = \frac{1}{1 + e^{\Delta E}/T}$$

Hence it is possible to move away from local optima. After searching sufficiently long, an equilibrium probability distribution over states of the search space is reached in which the probability of a state, α, is exponentially related to its energy:

$$\Pr(\alpha) = \frac{e^{-E_\alpha/T}}{\sum_\beta e^{-E_\beta/T}}$$

where T is a parameter analogous to temperature in a physical system. It determines how sharply the probability of accepting a move drops off as the change in energy goes from slightly negative to slightly positive. This distribution is known as the Boltzmann or Gibbs distribution. At high temperatures equilibrium can be reached quickly, but the resulting distribution is not very discriminating. At sufficiently low temperature only global optima have significant equilibrium probabilities, but reaching equilibrium is slow. By beginning the search at high temperature, and then gradually lowering it, it is often possible to find very good states in reasonable time.

3.6 Family Relations Problem

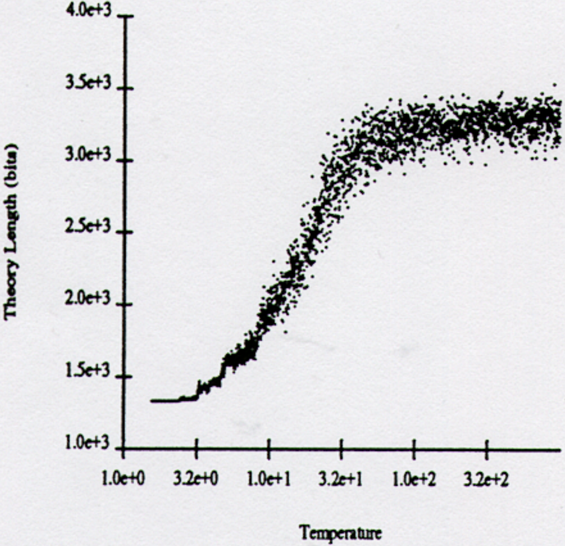
Table 1 shows that the evaluation function ranks combinations of intuitively reasonable features in a reasonable way. Figure 13 shows that it is sufficiently smooth that simulated annealing can find solutions as good as the presumed global optimum of {Person Sex Nationality Generation}. However it is prudent to examine the solutions actually found by the annealing search algorithm and verify that they are this desired set, or something else intuitive. Finding five binary features with the slowest annealing schedule shown in figure 13, the results obtained over 20 trials were as follows:

Feature	Frequency
Person	20
Sex	20
Nationality	20
Parent	19
Skewed-2-Way-Generation	7
2-Way-Generation	13
	99

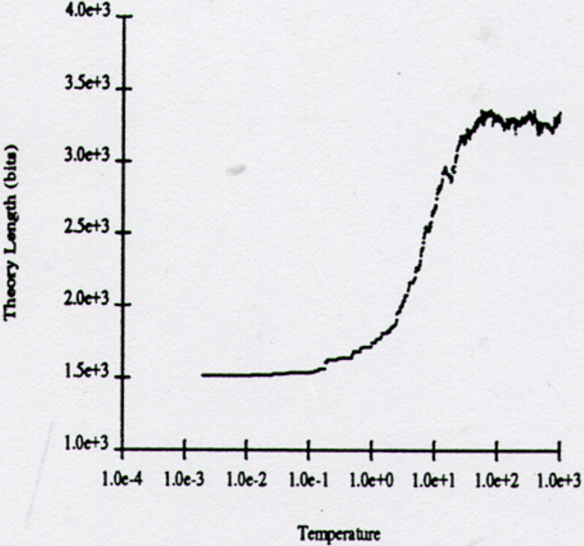
In words, it always found Person, Sex, Nationality, Parent (the unclassified feature was was a skewed version of Parent), and a version of 2-Way-Generation. The actual assignments for each of these features are shown in appendix D. In each case, symmetric versions of the features are lumped together. For instance, the Nationality category includes solutions where the relations are grouped either with the English or the Italians. For the next-slowest annealing schedule, 92 out of 100 solutions were one of these; another factor of 10 reduction in search time lead to only 56 of 100 solutions being one of these.

3.7 Scaling

There are q individuals and d features, so the search space size is $\prod_{i=1}^{d} c_i^q$. Each point has $q \sum_{i=1}^{d} c_i - 1$ neighbors. The time to calculate the change in evaluation function due to a move is proportional to the number of training examples in which the moving individual appears. Assuming individuals rarely appear multiple times in a training tuple, this can be approximated by nl/q. The number of moves necessary to find a good solution is difficult to estimate. It would seem to be at least linear in the number of individuals, the number of features, and their arities. I expect to seek a handful of binary features, independent of the problem size, and to always use $n = 3$. Hence the total search time might be expected to scale approximately as the number of training examples, l. It is hard to compare search time across domains, however, because the difficulty of finding the regularities in a domain is hard to quantify. The best approach to equating solution quality seems to be to adjust the annealing rate until it is just slow enough to give a smooth, non-monotonic energy versus temperature plot (see figure 14). Using this criterion, the largest problem I have tried requires between three and



4.0e+3
3.5e+3
3.0e+3
2.5e+3
2.0e+3
1.5e+3
1.0e+3
Theory Length (bits)
1.0e+0
3.2e+0
1.0e+1
3.2e+1
1.0e+2
3.2e+2
Temperature



4.0e+3
3.5e+3
3.0e+3
2.5e+3
2.0e+3
1.5e+3
1.0e+3
Theory Length (bits)
1.0e-4
1.0e-3
1.0e-2
1.0e-1
1.0e+0
1.0e+1
1.0e+2
1.0e+3
Temperature

Figure 14: The energy versus temperature graph on the left derives from sufficiently conservative parameters that it is smooth, yet non-monotonic. That on the right is too fast; at high temperatures it is jagged, while at lower temperatures it appears to do monotonic hill climbing. (Both plots include approximately the same number of points.)

four orders of magnitude more real time than the family relations problem, holding d and the c_i constant. This is much worse than the difference in training set size, which is only a factor of 30. However there are 200 times more individuals. Probably the number of moves required scales worse than linearly in q. If it were quadratic in q, this would account for the difference between the two domains.

Using only binary features decreases the size of the space drastically, and even in this case where generation is clearly a 3-valued feature, the solution is nearly as good. The search space can also be shrunk by finding only a few features and then "freezing" them while more are sought.

3.8 Prediction

Once the features are found, their regularities in the training set can be used for generalization to a test set. To do completion, a partial tuple such as (? husband Charles) is first mapped onto feature tuples, just as the MDL/OC encoder does. Mapping onto the nationality feature gives ?IE, for instance.[5] The nationality tuples which occur in the training set are EIE and III. Only EIE is compatible with this input, so the answer is expected to have the value English for the nationality feature. In general, there may be multiple possibilities, in which case all are recorded together with their relative frequency in the training set. For instance, mapping the sex feature gives ?MM, which matches two training set tuples, FMM and MMM. The first occurs 36 times, while the second occurs 20 times, so there is a 36% chance for the answer to be male. The only possible value for the generation feature of the answer is "1st Generation." Assembling all possible feature vectors for the answer, there is a 64% chance it is EF1 and a 36% chance it is EM1. The possible individuals with feature vector EF1 are {Penelope

[5] The feature-value abbreviations are defined in appendix D.

Christine} and the individuals with feature vector EM1 are {Christopher Andrew}. Each of these occur equally often (6 times) in the training set, so the completion algorithm assigns each of the women a probability of 32% and each of the men a probability of 18%. The completion results given in this paper choose the most probable answer, breaking ties arbitrarily. This algorithm shows that the feature tuples can be gathered at run time and used to do inference, but they aren't very intuitive as axioms. It would be nice to extract more general ones, such as "country of PERSON1 equals country of PERSON2." This is discussed in section 5.

3.9 Soybean Disease Problem

Given 290 soybean plant descriptions, each associated with one of 15 diagnoses, the system is to learn to diagnose the disease of test cases. Each description consists of 35 attributes, some of whose values may be unknown. MDL/OC maps each training tuple onto a set of feature-tuples, and must keep track of how many training tuples map onto each feature-tuple. In the current implementation each possible feature-tuple gets a unique index which is used as a hash key. The space of feature-tuples is exponential in the number of attributes, and maintaining this bookkeeping information becomes annoyingly slow because the indexes become bignums. The overhead of using bignum arithmetic slows down the algorithm by one or two orders of magnitude.

In addition to this practical problem, there is a theoretical reason MDL/OC won't do well on prediction tasks for arbitrary domains. It seeks regularities which can be captured by features. Any other information is encoded in the disambiguation information. While looking for only this simple kind of regularity aids the knowledge integration process, ignoring other kinds of regularities precludes it from learning a sufficiently complete domain theory except in special cases. Even in the family relations problem, the branch of family information is not sufficiently feature-like to be discovered.

Not surprisingly, therefore, MDL/OC isn't competitive for this problem. Starting with an initial temperature of 20,000, and decreasing it by a factor of .99999 after each potential move, a solution was obtained after about a day of CPU time on a Symbolics 3630. Performance on the training set[6] was

[6] These numbers are four four binary features. Using the completion model and the

69%, and that on a test set was, somewhat surprisingly, slightly higher: 74%. In contrast, others have obtained close to 100% generalization on the same test set [Michalski and Chilausky, 1980, Tan and Eshelman, 1988].

3.10 Country Trading Problem

MDL/OC is a data-hungry algorithm, while most of the CYC KB remains sparse. To get much compression there must be many assertions made about a few individuals. The part of the KB that is densest in this respect deals with geographical and political regions. For this learning task, the subdomain of countries and their trading behavior was chosen. This data was copied into CYC from a 1986 almanac and a 1988 almanac. The data involves 582 assertions making reference to only 19 countries and 8 slots, a very high density indeed. Before learning I had no idea what features would be found. Interpreting the features has been even more difficult than I imagined, and much better tools are required. In the limited time devoted to this data, only one feature not already in the KB has been intuitively understood. This feature segregates countries with large economies from those with small economies.

One tool which helps interpret features is a histogram of the resulting feature-tuples. Figure 15 shows the feature vector assignments for all six features, and Figure 16shows the corresponding histograms. The first feature distinguishes Countries from Slots. The histogram confirms that this is a useful feature. One third of the training set occurrences (the slots) have the value 1 for this feature, and two thirds have the value 0. If the training tuples were chosen randomly from this distribution, the expected number of 1-0-0 feature-tuple patterns would be $\frac{1}{3} \cdot \frac{2}{3} \cdot \frac{2}{3} \cdot 582 = 86$ (broken line), but the actual observed number is 582 (solid line). None of the other feature-tuple patterns are ever observed. The feature-tuple entropy of this feature is therefore zero, so it contributes nothing to the data term. Yet it contributes $-3 \cdot (\frac{1}{3} \log \frac{1}{3} + \frac{2}{3} \log \frac{2}{3}) = 2.75$ bits of information about the identity of each input tuple.

The second and third features also map all the training tuples onto a single feature-tuple pattern, 111, but this is only because all individuals have

:Person-2-only, :used codes variation as described in appendix ?? gave essentially the same results.

the same value for this partition. So the expected number is also 582, and the information contribution is zero. The fourth feature is easier to reverse-engineer from the table than the histograms, because it (nearly) corresponds to the distinction already in the KB between subabstractions of countries during 1986 and subabstractions of countries during 1988. CANADA-1988 does not fit this pattern, however. A quick glance through the training data shows that it always trades with the 1986 subabstraction of other countries. Presumably someone was careless in entering this unit and typed the wrong dates. This points out a less ambitous use for MDL/OC, finding oversights.

The fifth feature is the novel one I have made sense of. Countries with the value 0 have large economies, and those with the value 1 have small economies. Looking at the countries in the table, this accords with real-world knowledge reasonably well, except for the case of JAPAN-1988. I presume this also reflects shortcomings in the KB data. The feature is clearer with respect to the slots. Each "primary slot" is a relation "from the point of view of" its first argument. The corresponding inverse slots (which all end in "OF") are therefore from the point of view of the second argument. Assuming everyone trades with everyone, it will be those countries with larger economies that are major anythings. So for the primary slots, the second argument will usually be filled by a country with a large economy, and for inverse slots the first argument usually will be. This pattern is borne out in the histograms: $0x1$ and $11x$ hardly ever occur.

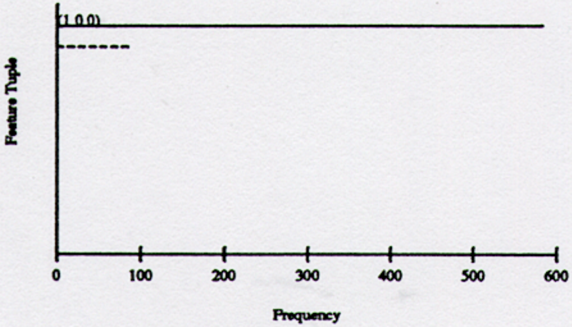
I have not been able to interpret the sixth feature. From the histograms it appears to be less significant, because the solid and dashed lines are in reasonable agreement. A more direct measure of the significance is the value of the evaluation function for each feature considered in isolation: (The value for all the features together is 4222 bits.)

Feature	1	2	3	4	5	6
Evaluation Function (bits)	4465	5559	5559	5444	5319	5535

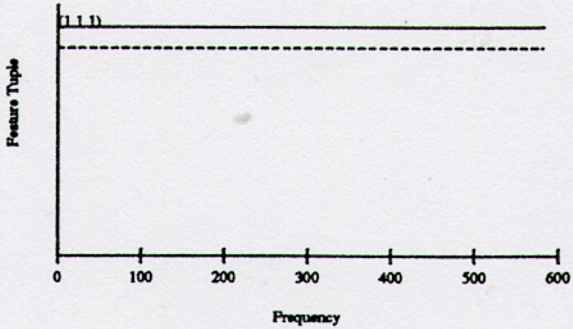
This exercise in interpretation illustrates how blind the process is. On the family relations algorithm there was the luxury of knowing the "correct" solution, so comparing algorithms was easy. On wild data it is imperative that the algorithm be parameterless and automatic. The two parameters of MDL/OC, search space size and annealing schedule, are fairly innocuous, because their relationship to performance is straightforward. The algorithm

Individual	Feature Vector					
JAPAN-1986	0	1	1	0	0	1
JAPAN-1988	0	1	1	1	1	0
ITALY-1986	0	1	1	0	0	0
ITALY-1988	0	1	1	1	0	0
SPAIN-1986	0	1	1	0	1	1
SPAIN-1988	0	1	1	1	1	1
FRANCE-1986	0	1	1	0	0	1
FRANCE-1988	0	1	1	1	0	1
WESTGERMANY-1986	0	1	1	0	0	1
WESTGERMANY-1988	0	1	1	1	0	1
NETHERLANDS-1986	0	1	1	0	0	1
UNITEDSTATES-1986	0	1	1	0	0	1
SAUDIARABIA-1986	0	1	1	0	1	0
AUSTRALIA-1986	0	1	1	0	1	0
PORTUGAL-1986	0	1	1	0	0	0
BELGIUM-1986	0	1	1	0	0	0
UNITEDKINGDOM-1986	0	1	1	0	0	1
CANADA-1988	0	1	1	0	1	1
NIGERIA-1986	0	1	1	0	1	1
MYMAJORTRADINGPARTNERS	1	1	1	0	0	1
MAJORTRADINGPARTNERSOF	1	1	1	0	1	0
MYMAJOREXPORTRECEIVERS	1	1	1	0	0	1
MAJORFOREIGNRECEIVEROF	1	1	1	0	1	1
MAJORFOREIGNMARKETS	1	1	1	0	0	0
MAJORFOREIGNMARKETSOF	1	1	1	0	1	1
MYMAJORIMPORTSUPPLIERS	1	1	1	0	0	1
MAJORFOREIGNSUPPLIEROF	1	1	1	0	1	1

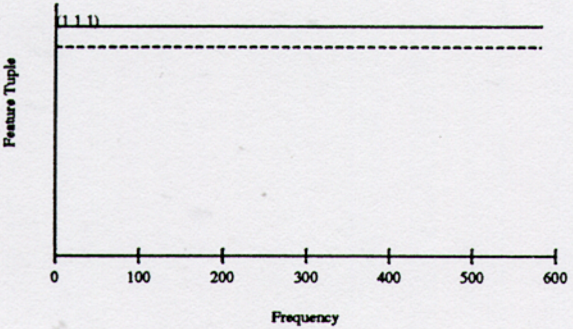
Figure 15: Features assignments discovered in one run over the country data.



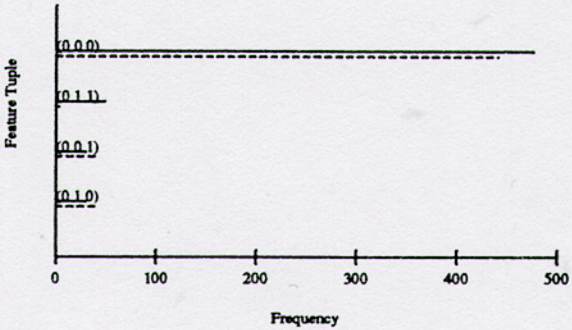
Feature 1: Slot/Country



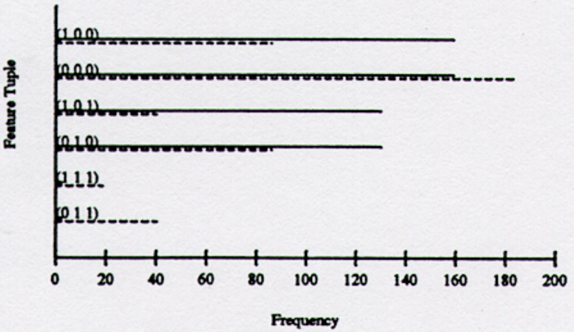
Feature 2: Uninformative



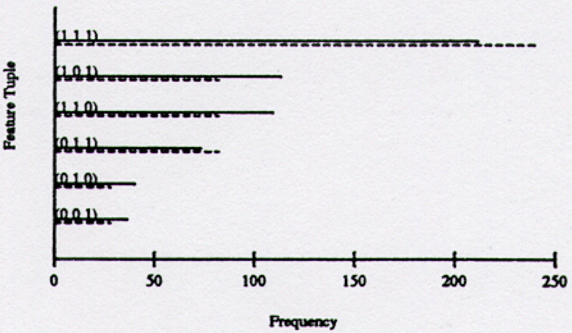
Feature 3: Uninformative



Feature 4: 1986/1988



Feature 5: Economy Size



Feature 6: ?

Figure 16: Histogram of feature-tuples for one of the six features discovered in one run over the country data. Solid lines are actual frequencies; dashed lines are expected frequencies given the unconditional feature probabilities and assuming independence across components.

can automatically choose a starting and stopping temperature. The length of the schedule can be gradually increased until the solution stops improving

4 Analogical Reasoning

4.1 MDL/OC Considered Analogical

MDL/OC contributes to three stages of analogical reasoning—selection, mapping, and applying the mapping to answer questions. In addition there is a simple MDL algorithm specifically for finding analogical mappings. Analogical reasoning was actually described in section 3.8, but because that algorithm does not explicitly find the maps, its analogical character may not have been obvious.

It is more apparent in an example involving very different domains, such as music and geopolitics. It may happen that the training data has no tuples matching (#%Drums #%maximumVolume ?). But perhaps #%Drums has similar features to #%UnitedStates, #%maximumVolume has similar features to #%belligerency, and #%VeryLoud has similar features to #%VeryBelligerent. Then the maximum-likelihood guess that #%Drums are #%VeryLoud can be explained by an analogical mapping. First the musical terms are mapped onto the geopolitical terms. Then facts known about that domain are accessed, and finally the relevant fact is mapped back. Of course none of this is explicit, but has all been "compiled" into the prediction function implicit in the feature-tuples. The feature-by-feature prediction solves one of the serious problems of reasoning by analogy, context. It is next to meaningless to say in isolation "x is analogous to y," where x and y are simple objects. One wants to find analogical mappings relevant to solving a particular problem [Greiner, 1988]. In this approach, relevant mappings pair individuals which share features relevant to the given relation. For instance if the relation is #%maximumVolume we want individuals with features highly predictive of features of that relationship. (Treating the relationship specially is only for ease of explanation. The feature-discovery criteria treat all components of a tuple symmetrically.) In a conventional analogy program, once the analogical mapping is established, the relevant facts must then be determined and mapped back. But here the prediction function automatically uses the appropriate relationships—it is not even necessary to find the analogical mapping explicitly because all the hard work was done off-line at the time the features were discovered.

In the mapping stage of an explicit analogical reasoning algorithm, a subset of the source domain predicates and individuals are associated with a subset of the target domain predicates and individuals. Taking Gentner's Structure Mapping Theory [Gentner, 1983, Falkenhainer *et al.*, 1989] as an example, there are some hard constraints on what mappings are considered. Of those that are allowed, all combinations are explicitly evaluated. So as the domains grow, there is a combinatorial explosion. Consequently it is impractical to map the target onto a whole KB. Further, if either domain contains irrelevant information, too much may be mapped. So a selection phase must precede the mapping, which isolates just the relevant information for mapping.

It is possible to use MDL/OC to find an explicit mapping. The search time grows linearly with the number of training examples, and hopefully not worse than this on the number of individuals. And the MDL principle eliminates irrelevant mappings no matter what their form. So the selection process need not be nearly so smart. A feature to differentiate individuals from the two domains is pre-assigned, and then MDL/OC is used to find additional features. Individuals with the same feature vectors (ignoring the built-in domain feature) are mapped together. So the mapping can be many to many.

Once the features are assigned, finding the map is very fast. If the features can be assigned independently of the particular pair of domains to be mapped, the expensive part can be done once and for all off line. Without an explicit built-in feature to differentiate the domains, more sophisticated similarity measures than identity of other features must be used. This problem of finding analogous individuals based on feature vectors has been explored previously. The approach outlined by Tversky [1977] seems especially attractive. An algorithm sensitive to the particular combination of source and target domain may well do a better job, though.

4.2 Direct MDL Mapping Algorithm

Using the features discovered by MDL/OC to find analogical mappings has at least two disadvantages. First, it is symmetric, whereas psychological evidence indicates that mappings found by people are asymmetric [Tversky, 1977]. Second, for small examples—the ones traditional analogical mapping algorithms work best on—MDL/OC may find very few features, so too many things are mapped together. A direct use of MDL avoids these problems.

In the MDL paradigm, the coding scheme must be tailored to what the receiver already knows. For analogical mapping he can be assumed to know not only what individuals are in both domains, but also all the source domain assertions. What must be transmitted are the target domain assertions. On the assumption that most of the source domain will be mapped, it will be efficient to transmit the mapping between source individuals and target individuals, because the receiver can then reconstruct most of the target domain just by mapping all the source domain assertions. Then any unmapped target assertions must be transmitted individually, as well as the index of any source assertions whose mapped version does not occur in the target domain.

This algorithm has been applied to a water flow/heat flow example that Falkenhainer *et. al.* [1989] use to illustrate their Structure Mapping Engine. In the water flow domain, the flow is from a beaker into a vial through a pipe. The cause of the flow is the fact that the pressure in the beaker is greater than the pressure in the vial. There are also two irrelevant facts, that the water has a flat-top and that the diameter of the beaker is greater than the diameter of the vial. In the heat flow domain, the flow is from some coffee to an ice cube through a bar. The temperature of the coffee is greater than the temperature of the ice cube, but there is no assertion that this causes the heat flow. There are also two irrelevant facts, that the coffee has a flat-top and is liquid. The SME syntax for expressing these assertions is shown in figure 17.

SME finds the following correspondences:

```
(defDescription simple-water-flow
  entities (water beaker vial pipe)
  expressions (((flow beaker vial water pipe) :name wflow)
               ((pressure beaker) :name pressure-beaker)
               ((pressure vial) :name pressure-vial)
               ((greater pressure-beaker pressure-vial) :name >pressure)
               ((greater (diameter beaker) (diameter vial)) :name >diameter)
               ((cause >pressure wflow) :name cause-flow)
               (flat-top water)
               (liquid water)))

(defDescription simple-heat-flow
  entities (coffee ice-cube bar heat)
  expressions (((flow coffee ice-cube heat bar) :name hflow)
               ((temperature coffee) :name temp-coffee)
               ((temperature ice-cube) :name temp-ice-cube)
               ((greater temp-coffee temp-ice-cube) :name >temperature)
               (flat-top coffee)
               (liquid coffee)))
```

Figure 17: Description of the liquid flow/heat flow analogy problem in the syntax accepted by SME.

(arg1 wflow flow)	↔	(arg1 hflow flow)
(arg2 wflow beaker)	↔	(arg2 hflow coffee)
(arg3 wflow vial)	↔	(arg3 hflow ice-cube)
(arg4 wflow water)	↔	(arg4 hflow heat)
(arg5 wflow pipe)	↔	(arg5 hflow bar)
(arg1 pressure-beaker pressure)	↔	(arg1 temperature-coffee temperature)
(arg2 pressure-beaker beaker)	↔	(arg2 temperature-coffee coffee)
(arg1 pressure-vial pressure)	↔	(arg1 temperature-ice-cube temperature)
(arg2 pressure-vial vial)	↔	(arg2 temperature-ice-cube ice-cube)
(arg1 >pressure greater)	↔	(arg1 >temperature greater)
(arg2 >pressure pressure-beaker)	↔	(arg2 >temperature temperature-coffee)
(arg3 >pressure pressure-vial)	↔	(arg3 >temperature temperature-ice-cube)
(arg1 diameter-beaker diameter)		(arg1 flat-top-coffee flat-top)
(arg2 diameter-beaker beaker)		(arg2 flat-top-coffee coffee)
(arg1 diameter-vial diameter)		(arg1 liquid-coffee liquid)
(arg2 diameter-vial vial)		(arg2 liquid-coffee coffee)
(arg1 >diameter greater)		
(arg2 >diameter diameter-beaker)		
(arg3 >diameter diameter-vial)		
(arg1 cause-flow cause)		
(arg2 cause-flow >pressure)		
(arg3 cause-flow wflow)		
(arg1 flat-top-water flat-top)		
(arg2 flat-top-water water)		
(arg1 liquid-water liquid)		
(arg2 liquid-water water)		

Figure 18: To force all the assertions to be tuples of the same arity, each named assertion in SME notation is reified, and its arguments are asserted using the generic relations arg1, arg2, etc. This is the same technique used to fit arbitrary assertions into frame languages, in which the only available type of relation is slots, which are binary.

>pressure	↔	>temperature
pressure-beaker	↔	temp-coffee
pressure-vial	↔	temp-ice-cube
wflow	↔	hflow
beaker	↔	coffee
vial	↔	ice-cube
water	↔	heat
pipe	↔	bar

In addition it hypothesizes that the cause of the heat flow is the temperature difference.

It is possible to apply the special purpose MDL analogical mapping algorithm to the water/heat flow problem as stated, but the MDL/OC algorithm requires that all tuples be of the same arity. The input was therefore rewritten as shown in figure 18 for use by both algorithms.

The special purpose algorithm finds exactly the same correspondences as SME. The logical way to find candidate inferences, such as that the temperature difference causes the heat flow, is to map some source assertions to the target domain. The causality relation is one that can be mapped back, but so is the assertion (flat-top water), which becomes (flat-top heat). Falkenhainer *et. al.*'s arguments that the relevant assertions can be determined syntactically are unconvincing. Rather, it seems that unless the selection process can eliminate irrelevant assertions, a syntactic mapping algorithm had best not make candidate inferences.

Because this algorithm uses such a simple syntax, not even distinguishing predicates from arguments, it has a greater range of freedom in finding analogies. For instance it can map a second-order description of a domain onto a first-order description of the same domain. Of course exploring a larger space of possible analogies will be unnecessarily slow if these further flung possibilities are rarely useful. It is possible to go even further than was done for the water flow/heat flow example. The domains could be "standardized apart" so they share no terms. It might be useful to separate water-flow-liquid from heat-flow-liquid since they play such different roles in each. It is probably best not to create copies of the most basic terms, like THING, ISA, and perhaps even CAUSE. It might also be useful to use different dummy arguments when transforming to a frame-based representation. For instance, if the water-flow domain included the assertion (less pressure-vial pressure-beaker) rather than (greater pressure-beaker pressure-vial), it becomes difficult to map onto the corresponding statement about temperature in the heat-flow domain because the difference in argument order obscures the systematicity. But if transformed to

(<pressure-arg1 <pressure less)
(<pressure-arg2 <pressure pressure-beaker)
(<pressure-arg3 <pressure pressure-vial)

the relation is just as easy to see as in the original version once the arg$x$ arguments are mapped as follows:

<pressure-arg1	↔	>temperature-arg1
<pressure-arg2	↔	>temperature-arg3
<pressure-arg3	↔	>temperature-arg2

5 Future Work

I see two principle directions in which to extend MDL/OC. The first is a direct continuation toward the original goal of facilitating knowledge integration. The second is to try to use its discoveries directly, without the intermediate step of integration.

5.1 Knowledge Integration

One obvious problem in the family relations domain is that the feature NATIONALITY is meaningless for relations. That is, there really is a hierarchical relation here. It is also straightforward to recognize this situation. If there exists a pair of features, N and P, such that, averaged over all components of the training tuples, the conditional mutual information of N about the other components of the tuple is negligible except for one value of P, then N should be hierarchically dependent on that value of P.

A second task to be further investigated is simplifying the problem from discovering classes *ab initio* to cleaning up and extending existing class assignments in the KB. This was already demonstrated in section 3.10 in which CANADA-1988 was clustered with subabstractions from 1986. It was then easy to notice the error that had been made in knowledge entry. Similarly, if a country had not been given an entry for the slot #%sameEndPoints, but was clustered with countries all of which had the value #%TheYear-1988 for this slot, it would be easier for a human to notice and add the missing information. In this case it was fortuitous that MDL/OC found an existing partition. It is also possible to build in a partition we would like to clean up, and let the algorithm hill-climb (or anneal from a relatively low temperature) to correct mistakes and extend the domain of applicability.

Doug Lenat (personal communication) suggested that it would also be very helpful to have a tool that will search the KB for existing common features of each newly-discovered partition element. One could hope that the fact that CANADA-1988 was misplaced could be discovered totally automatically, because the very simple characterization "countries with #%sameEndPoints #%TheYear-1988" correctly defines the individuals with that feature assignment, with this one exception.

A more difficult objective is to learn more sophisticated theories. Most clustering algorithms learn an intensional description of the *classes*. In fact, it is hard to see how they could generalize to new individuals without this characteristic. Unfortunately, MDL/OC only learns extensional lists assigning individuals to classes. Along with this, it is unable to generalize to new individuals, as discussed in section 2.1, because its training tuples are composed of individuals rather than attributes. But it should be possible to learn an intensional characterization of the *domain*—an axiomatic theory. The feature tuples constitute a rather extensional theory of which relations occur. But it would be nice to extract rules like "relatives always have the same nationality" from the two extensional nationality patterns: Italian-NIL-Italian and English-NIL-English. This task can also be approached within the MDL framework. The task then becomes optimally coding the feature-tuples. Three useful rule-schemata are

1. Set of components which must have the same value for this feature.

2. Set of components which have a particular constant value for this feature.

3. Set of components which are unconstrained by this feature.

These schemata suffice to define the rules for SEX, PERSON, and NATIONALITY. GENERATION seems too complicated to describe with such simple rules. The notion of an ordering on feature values would be required, at least.

Using a more complicated code for the feature-tuples facilitates reverse-engineering any discovered solution. Further, since intentional rules will be shorter, simple theories will be encouraged. Still, aids will be required for understanding the solutions. In the case of CYC, much background knowledge exists that should be called upon to make sense of the extensional lists of individuals. A general-purpose set-describer that could digest a list of countries and produce a description like "these are all coastal European countries with a Parliamentary form of government, except for Rhodesia" would be very useful. Devising a simple language for describing sets ought to be straightforward, and is enough to generate an MDL evaluation function.

5.2 Reasoning

Since traditional clustering algorithms seem to be better for pure prediction tasks, the most fruitful area for reasoning with MDL/OC appears to be finding and applying analogies, either implicitly or explicitly. During knowledge entry it is very useful to be able to find similar individuals, both to avoid possible duplication and as a source of inspiration. CYC currently does this by finding units which share values for many slots. By using the features discovered by MDL/OC, this same technique can be used over primitives at a higher level of abstraction.

The special-purpose analogical mapping algorithm can be called when a pair of domains ripe for analogical transfer is known. This may actually become quite common as CYC makes wider and wider use of micro-theories. Much work will have to go into rules for importing assertions made in one context for use in another, slightly different one. Perhaps many of the easy cases can be taken care of automatically.

Finally, there is hope that MDL/OC could actually surpass other algorithms at prediction if applied to heterogenous domains which become too fragmented in a hierarchical decomposition. This is where implicit analogical reasoning would be most helpful.

6 Conclusion

This paper has described a new approach to unsupervised clustering, called *orthogonal clustering*, and described its strengths and weaknesses. It is based on a well-motivated declarative description of what good clusters are. The only subjectivity which entered in the derivation is the decision to use an MDL approach at all, and the theory-language in which to minimize. The resulting expression has no parameters to adjust. Actually finding solutions which optimize the expression, however, is an intractable problem. The search algorithm used in this paper, simulated annealing, does require empirical parameter setting to work well, and the search is slow. Although scaling was briefly examined, more experience with real-life problems will be necessary to evaluate whether good solutions can be found in practice.

A somewhat surprising result has been the realization that completion performance on a test set is not a very good measure of the intuitiveness of clusters for the orthogonal clustering problem. Occam's MDL razor is a more appealing *a priori* way to prevent overfitting than the *post hoc* stopping or pruning criteria using multiple partitions of the data into training set, stop training set, and test set. Using orthogonal, as opposed to hierarchical, clustering also prevents data fragmentation. Near the leaves, decision tree learning algorithms, for instance, may have so few training instances that observed statistical regularities are almost entirely spurrious. Guilia Pagallo and David Haussler [1990] suggest a way to recombine identically structured subtrees after the tree is learned. This can be thought of as orthogonalizing.

In any case, very few of the "symbolic" clustering algorithms have been applied to tuples of individuals. Of those that do (see appendices A-B), MDL/OC generates a more intuitive ranking of features for the family relations problem.

Hierarchical clustering finds context-specific regularities, while orthogonal clustering finds independent regularities over all the data. Hence it should be ideal for analogical reasoning, which after all is just extending regularities beyond their customary context. It is appealing that in this sense, doing completion with MDL/OC is doing analogical reasoning without explicitly going through the several stages customary in the literature. Even if the vast majority of reasoning using far-flung knowledge is automatic, however, there is a need for explicit conscious analogical reasoning, and for finding analogical mappings. The suggestions for doing this using MDL/OC above were very weak methods. Although other analogy algorithms are domain independent to the extent of being entirely syntactic, such as SME, MDL/OC has such a weak syntax that it seems a step more extreme than any others. Knowing only about tuples, it has no need to distinguish predicate from object, little less characteristics like which predicates take predicates as arguments. While this extreme weakness and generality is appealing, it will probably have to be augmented with more knowledge to be effective.

Acknowledgements

I am indebted to Geoff Hinton for advice and encouragement through several active stages of my obsession with the family relations problem, and to Wei-Min Shen for active support during this last stage reported here. I am also grateful for suggestions about MDL/OC and tutoring in machine learning to David Aha, Sue Becker, Peter Brown, Guha, Eric Hartman, Jim Keeler, Doug Lenat, Ken Murray, and Jeff Rickel.

Appendices

A Hinton's Algorithm

The quest for an orthogonal clustering algorithm described in this paper began by tweaking Hinton's back-propagation approach to the problem. So in this section both his algorithm and an analysis of the results are given.

Rather than have a declarative evaluation function over sets of clusters, Hinton used the generalization performance realized by the clusters on a completion task as a measure of their worth. He only requires the network to be able to fill in the PERSON2 component of a triple, given the PERSON1 and RELATION components.

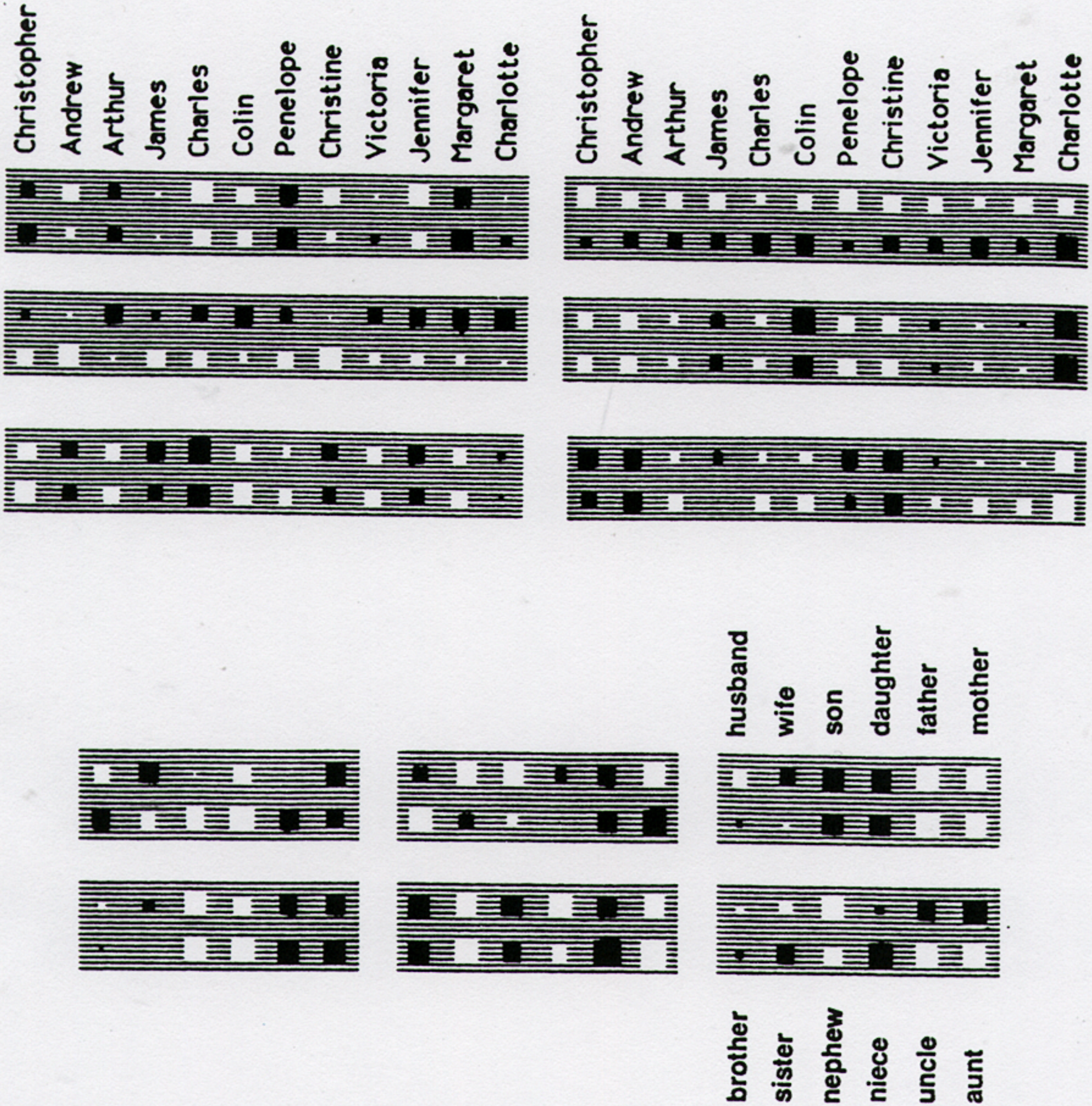
Hinton argues that if the network represents the data in terms of *micro-inferences* among features of the individuals, rather than at the level the input is presented, the domain theory is much simpler. For instance the micro-inference that relatives are always of the same nationality is very simple to state, and rules out half the possible answers to any question. A side-effect of having a concise theory is increased ability to generalize. If the network learned the training corpus by rote, it could not plausibly guess who Penelope's husband is unless it could find a matching training instance. But by using micro-inferences it can conclude that the answer is someone of the same nationality, the same generation, the opposite sex, and the same branch of the family, namely Christopher. The way Hinton shows that the network in fact "understands" the domain as opposed to having learned the training set by rote is its ability to generalize.

Hinton uses a unary representation for each of the two input components, so the PERSON1 group in figure 19 has 24 units and the RELATION group has 12. To present the training instance (Penelope husband Christopher), the Penelope unit in group PERSON1 is turned on, as is the husband unit in group RELATION. Forward propagation takes place through the network, and the units in group PERSON2 take on values between zero and one. This group also has 24 units, one for each person. The network's answer to the question is derived by thresholding the unit states at 0.5. That is, for all the units whose state is greater than one half, the corresponding person is said to be (one of) Penelope's husbands.

The network is initialized with random weights, so the output states

```
                   xxxxxxxxxxxx  PERSON2
                   xxxxxxxxxxxx
                         |
                      xxxxxx  PERSON2-FEATURES
                         |
                   xxxxxxxxxxxx  CENTRAL-GROUP
                      /     \
PERSON1-FEATURES  xxxxxx  xxxxxx  RELATION-FEATURES
                    /         \
     PERSON1  xxxxxxxxxxxx   xxxxxx  RELATION
              xxxxxxxxxxxx   xxxxxx
```

Figure 19: Hinton's network architecture for solving the family relations problem. Each "x" represents one connectionist unit. Lines between groups of units indicate complete interconnection between all units in the pair of groups. PERSON1 and RELATION are input groups, PERSON2 is an output group, and the remainder are hidden groups. The receptive field developed in each unit of the PERSON1-FEATURES and RELATION-FEATURES (shown in figure 20) groups can be interpreted as representing a partition of the domain.



Christopher
Andrew
Arthur
James
Charles
Colin
Penelope
Christine
Victoria
Jennifer
Margaret
Charlotte
Christopher
Andrew
Arthur
James
Charles
Colin
Penelope
Christine
Victoria
Jennifer
Margaret
Charlotte
husband
wife
son
daughter
father
mother
brother
sister
nephew
niece
uncle
aunt

Figure 20: Hinton's solution to the family relations problem. Each gray rectangle shows the receptive field of a single back propagation unit. The six units in the PERSON1-FEATURE group are shown on top, and the six RELATION-FEATURE units are on the bottom. Black blobs stand for inhibitory connections; white blobs stand for excitatory connections. Blob sizes indicate the magnitude of the inhibition or excitation. The bottom row of PERSON1-FEATURE blobs represent the connections to the Italians. The labels for this row are not shown, but isomorphic individuals are paired vertically.

are initially also random. Learning takes place by repeated presentation of all training instances. On each presentation an input pair is clampled, generating a vector of output states. This vector is compared to the desired vector, in which the units corresponding to correct individuals have a state of 1.0, and other have state 0.0. The error is computed as the sum of the squares of the component-wise differences of the vectors, and its derivative is computed with respect to each weight in the network. Then the weight values are changed by a small fraction of the derivative. Hinton trained the network on a randomly chosen subset of 100 of the possible 104 questions from this domain until the error was very small, and then tested it on the remaining four. In one trial all four were answered correctly, while on another trial from different random initial weights it correctly answered three test questions. More importantly, when he examined the receptive fields of hidden units, he found some that came on if and only if PERSON1 is male, some that came on if and only if RELATION required that PERSON1 and PERSON2 be of the same generation, and so forth.

But there are some drawbacks to this indirect approach of using a question answering task when the real interest is the internal representations. First, back-propagation is deterministic, so if asked who is Colin's uncle it will always give the same answer. One of Colin's uncles would necessarily remain secret in a straightforward encoding in which each training example corresponds to a possible input/output mapping. Hinton's solution is to combine all training instances whose "question part" is the same by unioning the "answers." This introduces an asymmetry in the representation in that for the subject and relation, there can only be one unit on, while for PERSON2 there can be multiple units on. From the point of view of answering questions about PERSON2 this makes sense, but from the point of view of assigning features to individuals it is unnatural. A second artifactual asymmetry results from distinguishing inputs and outputs; the network only has an incentive to learn features which help encode regularities affecting PERSON2.

A second drawback is that the crucial notion of simplicity of the resulting domain theory is enforced as a hard constraint rather than as an explicit term to minimize as in MDL/OC. Architecturally, Hinton assigned one "feature group" to each input or output group, hoping that units for sex, nationality, generation, and branch of the family would develop there.[7] By making the only communication route between PERSON1 and the rest of the network be through a group with only six units, the network must somehow compress the 24-valued unary representation used in group PERSON1. This in fact works—the network does use a compressed representation and most of the units are clear enough to be interpreted in intuitive terms like nationality, at least if the experimenter knows the "correct" results to look for (see figure 20).

But left to their own devices, back-propagation networks using a least-squares error measure form very diffuse representations. So the network is fighting the narrow bottlenecks imposed by the feature groups. Hinton gave it six units and that's how many it used. Intuitively there are four features in this domain, and it would be better if the network could determine this for itself. Features with fewer numbers of values are also intuitively better. The network seems to prefer this as well. It could, after all, have used a single 24-value feature.

In addition to minimizing the number and arities of the features, another aspect of the intuitive notion of theory simplicity is the regularity of the tuples of features induced by the tuples of individuals. Since sex is a two-valued feature, there are eight possible tuples over the three components of each training instance. But only four actually occur, because the sex of the RELATION perfectly determines the sex of PERSON2. And this is completely independent of the values of the other features. To take an exteme case of irregular features that nevertheless suffice to solve the problem, and fit the six-unit bottleneck constraint, consider the binary representation of the order of an individual in a random permutation. If there are q individuals, the size of the representation will be $d = \log_2 q$. For each individual, let the value of the dth feature be the dth bit in its binary representation. Since $\log_2 24 < 6$ this representation fits Hinton's architecture, and it is sufficient to reconstruct the individual in the PERSON1 group, so perfect question answering is possible. The fact that the network develops a more intuitive set of features is due to two sources of bias in the network toward "simple" functions. First, the size of the central hidden group is limited, so the complexity of the function mapping from input feature vectors to output feature

[7] Since Hinton learned role-specific weights, there is no need to explicitly represent the feature PERSON.

vectors has a hard constraint. Second, within these hard limits on the functions computable by a given network, the simple ones generally involve fewer and less precise constraints between cooperating weights. So the network is more likely to chance upon a simple solution.

Hinton's architecture seems to have done a good job of creating features with few values and simple mappings, but not such a good job on limiting the number of features. In appendix B an analytical expression is derived for what constitutes a good set of features, so it is not necessary to depend on the built-in bias of back-propagation.

B Interdependence-based Algorithm

More recently, Hinton, Sue Becker, and Conrad Galland [Becker and Hinton, 1989, Galland and Hinton, 1990] have used an information-theoretic evaluation function to learn features using unsupervised back-propagation. This removes the asymmetry evident in Hinton's back-propagation network and makes explicit one of the desired characteristics of good features. This section describes my attempt to extend their work to learn multiple features simultaneously, so it could be applied to the family relations problem. At IBM, a group using an information theoretic approach to speech recognition has applied similar ideas to learning multiple features of English letters [Lucassen, 1983].

To repeat from section 2.2, I believe intuitively good features should have three characteristics. First, they should retain enough information about an individual that predictions about other components of a tuple can be made. Second, this prediction should be simple. Finally, a good set of features should not be redundant. Both the work at Toronto and at IBM, as well as the similar algorithm described below, address prediction by explicit optimization. But only the algorithm presented here addresses the other characteristics, and it addresses simplicity by imposing a hard, not entirely appropriate, bound, rather than by optimization. MDL is more satisfying in that it combines all three characteristics in a single expression to optimize.

B.1 More Information Theory

The *mutual information* between $\mathcal{U}$ and $\mathcal{S}$ is a symmetic measure of the difference between individual and joint entropies: $I(\mathcal{U},\mathcal{S}) = H(\mathcal{U}) + H(\mathcal{S}) - H(\mathcal{U},\mathcal{S})$. If $\mathcal{U}$ and $\mathcal{S}$ tell nothing about each other, $I(\mathcal{U},\mathcal{S}) = 0$. If $\mathcal{U}$ completely determines $\mathcal{S}$, $I(\mathcal{U},\mathcal{S})$ achieves its maximum, $H(\mathcal{U})$. Thus mutual information is a good measure for how much knowing the nationality of PERSON1 tells about the nationality of PERSON2. Good features have high mutual information across the training tuples.

Mutual information is only defined over pairs of distributions, however. In order to measure the predictivity across tuples with an arbitrary number of components, the more general concept of interdependence [Watanabe, 1969] is useful:

$$J(\mathcal{U}_1,\mathcal{U}_2,\ldots,\mathcal{U}_d) = \sum_{i=1}^{d} H(\mathcal{U}_i) - H(\mathcal{U}_1,\mathcal{U}_2,\ldots,\mathcal{U}_d)$$

J is also non-negative, and it reduces to mutual information when $d = 2$. Besides being a generalization of mutual information, the definition of interdependence can also be motivated by the cross-entropy function which measures the "distance" between probability distributions defined over the same events, A.

$$G(\mathcal{V},\mathcal{W}) = \sum_{\alpha\in A} \Pr(\mathcal{V}=\alpha)\cdot\log\frac{\Pr(\mathcal{V}=\alpha)}{\Pr(\mathcal{W}=\alpha)}$$

G is termed a distance because it is non-negative, is zero only for identical distributions, is symmetric, and obeys the triangle inequality. If α ranges over the joint events of $\mathcal{U}_1,\mathcal{U}_2,\ldots,\mathcal{U}_d$, $\mathcal{V}$ is the joint distribution of $\mathcal{U}_1,\mathcal{U}_2,\ldots,\mathcal{U}_d$, and $\mathcal{W}$ is the joint distribution of $\mathcal{U}_1,\mathcal{U}_2,\ldots,\mathcal{U}_d$ assuming independence (that is, $\mathcal{W}$ is the product of the marginal distributions), then $J(\mathcal{U}_1,\mathcal{U}_2,\ldots,\mathcal{U}_d) = G(\mathcal{V},\mathcal{W})$. In other words, maximum interdependence occurs among distributions for which the assumption of independence fails most definitively.

B.2 Evaluation Function

There are many ways to formalize the three goals of predictivity, simplicity, and irredundancy. For instance maximal predictivity is achieved in a probabilistic sense if the mutual information that the features of the known components convey about the unknown components is maximized. Similarly, redundancy can be measured by the interdependence of the known features. The simplicity of the prediction function might be measured using its Kolmogorov complexity.

One drawback of having three separate measures is having to combine them. Is one bit of predictivity worth one of Kolmogorov complexity? An alternative to simultaneously optimizing all three criteria is to optimize a subset of them under an *a priori* bound on the rest, as Hinton's back-propagation network did. Perhaps one could impose requirements that predictivity be maximized absolutely; that redundancy be minimal over the maximally predictive features; and the complexity be minimized over these. In this section complexity is bounded absolutely, and within this bound predictivity and redundancy are traded off, giving each equal weight. As will be seen below, these measures are naturally commensurable: the optimization corresponds to minimizing an interdependence measure in the interior of a tree, while simultaneously maximizing it at the leaves. The complexity bound is simple: each feature predicts only the value of that same feature for the unknown components. To the extent that sex of PERSON1 predicts the sex of PERSON2, the evaluation function is rewarded. But it gets no credit for any dependence of the branch of the family of PERSON2 on the sex of PERSON1. Indeed there is a penalty.

The interdependence[8] across components of feature f_i is just $J(f_i)$, so the predictive half of the evaluation function to be minimized is $-\sum_{i=1}^{d} J(f_i)$. This builds in the restriction on the complexity of the prediction function, because $J(f_i)$ only measures the predictivity associated with f_i. The interdependence across features is $J(f)$. Summing these, we minimize $E(f) = J(f) - \sum_{i=1}^{d} J(f_i)$. The resemblance of this to the expression for $J(f)$ itself is uncanny. Expanding it out into entropy terms, $E(f) = -H(f) + 2 \cdot \sum_i H(f_i) - \sum_{ij} H(f_{i_j})$.

It might seem that the definition of redundancy as inter-feature interdependence leaves something out. It is undesirable to have a huge set of features each of which makes a prediction for a small fraction of the training set, even if they have low interdependence and collectively have high

[8] In this section, the only random variable used is $\mathcal{S}$, so for simplicity it will not be mentioned explicitly. For instance $H(f_i)$ means $H(f_i(\mathcal{S}))$.

predictivity. But a little thought shows that this cannot happen. Imagine that $d-1$ non-interdependent features have been found, and we are trying the find the dth non-interdependent one. Consider only the interdependence that results from the probability distribution over a single component of the training tuples. (If the dth feature is to have no interdependence, it surely can't have any intra-tuple interdependence.) To achieve this, its partition of the individuals must be orthogonal to those of the previous features. Equivalently, given the fraction of individuals that take on each value for this new feature, each element of the cross-product of the other partitions must be divided up in the same ratio. If the smallest of these has only one element, there can be no non-trivial non-interdependent new feature. Any set of d non-interdependent features must therefore satisfy $\prod_{i=1}^{d} c_i \leq q$. The maximum is achieved if all features represent even binary features, in which case $d = \log_2 q$.

In the case where we do not demand absolute non-interdependence, the global minimum of the evaluation function can of course occur for a larger number of features. However $J(f)$ will be positive, and thus it must be the case that greater intra-feature predictivity has been achieved than if absolute non-interdependence had been required.

The real problem is just the opposite. If unconstrained, this evaluation function will find too few features, each with too many possible values. It is most happy with a single q-ary feature, which has no inter-feature interdependence, and optimal intra-feature interdependence. A hard limit is therefore placed on the maximum arity allowed any feature. This is quite natural in the algorithms discussed below.

Although good results were obtained on the family relations problem using discrete simulated annealing in the interdependence evaluation function, it has several drawbacks. There is no principled way to weight the three constraints of predictivity, simplicity, and irredundancy. *A priori* limits must be placed on the feature arity, or the result will be the identity feature. The Minimum Description Length (MDL) Principle results in a single parameter-free error function which gives intuitive values for all numbers and arities of features.

B.3 Search Algorithms

The interdependence evaluation function has the same kind of ingredients as the MDL evaluation function. The same discrete simulated annealing search algorithm seems to work best. Some alternatives are described here for completeness. All except those based on polychotomic trees are applicable to the MDL evaluation function as well.

B.3.1 Direct Approach

Back Propagation Since Hinton had been successful with back-propagation both in his family relations paper and in the later work with a mutual information evaluation function, that is the first approach I tried. The architecture is shown in figure 21. There is only one layer of weights, which connects the input units to the feature units. Corresponding weights for different components of the 3-tuple were tied together[9] so there would be a single function f. For each training tuple, one input unit for each component was set to 1.0, and all other units were set to 0.0. The state of the feature units was determined by applying the sigmoid function to the weighted sum of states from the inputs, just as in regular back-propagation. However the error was derived from E, which requires no supervision. Each unit's output state, which is guaranteed to be between zero and one, is interpreted as an independent probability that the corresponding binary feature takes on the value 1. This determines a probability distribution over all possible tuples of feature vectors for a given training tuple, and hence also determines an average distribution over all the training tuples. In order to back-propagate, we need the derivative of the evaluation function with respect to the state of each feature unit, but we will not bother to write this out.

This architecture requires that each feature unit take on as many states as there are possible feature values. In the case discussed above there are two possible values, but we are using continuous-valued units. A conceptually more precise account is that the output units are binary and stochastic. This makes the network a Boltzmann Machine [Hinton and Sejnowski, 1986]. How-

[9] Since people never appeared in the RELATION component, and vice versa, there was no need to simulate all 36 possible input units for each component. Hence there are no other weights corresponding to relation inputs to tie together. Only pairs of weights for the PERSON1 and PERSON2 components were actually tied together.

```
PERSON1-FEATURES    RELATION-FEATURES    PERSON2-FEATURES

     xxxxxx              xxxxxx               xxxxxx
        |                   |                    |
  xxxxxxxxxxxx           xxxxxx            xxxxxxxxxxxx
  xxxxxxxxxxxx           xxxxxx            xxxxxxxxxxxx

    PERSON1             RELATION              PERSON2
```

Figure 21: Architecture for the unsupervised back-propagation network. In comparison to Hinton's architecture (figure 19) the PERSON1, RELATION, and PERSON2 groups serve the same functions, as do the corresponding feature groups. However without the necessity for supervision, this network does not require the central hidden group to implement the prediction function, nor the asymmetry in which PERSON1 and RELATION are dedicated inputs and PERSON2 is a dedicated output.

ever to do learning with a Boltzmann Machine requires many trials in order to average out the statistical noise in the units' decision rule. One technique for speeding up Boltzmann learning is to use a mean-field approximation [Peterson and Anderson, 1987], in which the units have continuous outputs which are interpreted as probabilities. The drawback of this is that higher-order correlations among unit states are lost. In this architecture, however, the Mean Field Approximation is exact, because no pairs of unclamped units are interconnected. This justifies the probabilistic interpretation of the states of the feature units. In order to learn features with more than two values, we can use Potts units [Peterson and Soderberg, 1989]. Alternatively, we can use continuous-valued units directly to represent continuous features. In the family relations domain this would allow features like #%age. For the case of learning a single feature at a time, Becker and Hinton [1989] have applied both the binary and continuous models to learning to recognize depth from one-dimensional random-dot stereograms. That work was a direct inspiration for the work reported here.

Even for the special case of trying to find a single binary feature from the family relations data, however, back-propagation was not very successful.

Computing the derivative requires time exponential in the number of features, since we must sum over all possible feature-vector tuples.[10] Further, starting with small random initial weights and proceeding in quite small steps either in the direction of the gradient or using momentum, the system always ended up in a poor local optimum. Perhaps more sophisticated techniques, such as using the second derivatives, would work better.[11]

B.3.2 Successive Refinement

Polychotomic Trees Although the search time seems to scale reasonably with the number of features, eventually it will be desirable to find sufficiently many features that a simultaneous search is impractical. An elegant decomposition property of the interdependence error function allows leads to several variations on a greedy algorithm for finding features sequentially.

Watanabe discusses a way to decompose a system into parts that he calls *multiplicative polychotomic trees*. Figure 22 is an example of such a tree, in which branch points represent orthogonal partitions of the system. The term "multiplicative" indicates that the set of states of the subsystem represented by any node is the cross product of its children's set of states. The family relations features are of this kind, because being male and Italian are independent events rather than mutually exclusive alternatives. In the cases Watanabe considers, the leaves correspond to specified atomic components. For now, therefore, we assume there is a fixed set of desired features, and we consider alternate derivation trees leading to them. Watanabe defines the total interdependence, J_{tot} of a system as the sum of the interdependences in a multiplicative polychotomic tree that divides it into its atomic constituents. As figure 22 shows, J_{tot} is independent of the structure of the tree.

The simultaneous feature-finding algorithms described above correspond to a polychotomic tree with only a single level of internal nodes below the root, which represent the features. We purposely treat atomic constituents representing different components of the same feature as a unit, not allowing them to be separated until the the lowest branching points in the tree. Let's call these branching points the "lower tree," and the other branching points the "upper tree." Then the evaluation function is optimized when the

[10] This isn't as bad as it sounds, because as argued in section B.2 the number of features is only on the order of the log of the number of individuals.

[11] Hinton, personal communication.

interdependence of the upper tree is minimized, and that of the lower tree in maximized. Since the total interdependence is independent of the tree structure, and we have specified that only feature-to-component branches may occupy the lower tree, the value of the evaluation function is also independent of the tree structure. This suggests we can derive the features via $d-1$ binary branches rather than one d-ary one.

In reality the final features are not specified at the outset. However for any tree which leads to an optimal feature-set, all upper branch points will have low interdependence. Therefore a good heuristic is to iteratively find branch points with minimal interdependence. Further by ensuring at each stage that distinct individuals are kept distinct in at least one of the branches, then $J_{tot} \geq H(f_0)$, where f_0 is a $q$ary feature which assigns each individual a unique value. The difference between this and the sum of the interdependences at the branch points at least provides a lower bound on the predictivity of the features.

The most natural way to gradually build up the polychotomic tree is successive even divisions. If there are q individuals originally, we first find two orthogonal partitions of arity $\sqrt{q}$. The resultant partitions are very coarse features, each of which can be further refined. The process can be stopped as soon as the arities seem reasonable. Here we arbitrarily stopped when the arities became either 2 or 3.[12] On these stopping (leaf) nodes, the function to be minimized is $E(f)$ (on the mapped-down problem), while for interior nodes it is $J(f)$. The results are terrible—the mean, -0.16, is only slightly better than zero, which can be achieved by a completely non-informative feature-set in which every individual has the same feature-vector. The algorithm is too short-sighted, only reducing interdependence at each interior node, without looking ahead to see that the leaf nodes will have high interdependence. One possible heuristic to avoid this problem is to minimize $E(f)$ at all nodes, interior and leaf, in order to encourage predictivity for each coarse feature. This worked much better, as the mean improves to -1.42. The supposed global optimum has a value of -2.56.

With this successive refinement algorithm, there is no guarantee that minimizing interdependence in the tree interior will leave highly-interdependent features at the leaves. Although the results for symmetric successive refinement were poor, we have had more success with very asymmetric refinements.

[12] It always helps to know the solution in advance!

```
        n&s&g                 Jn,s,g = Hn + Hs,g - Hn,s,g
         / \
       /     \
     /         \
    n           s&g             Js,g = Hs + Hg - Hs,g
  / | \         / \               Jn = Hn1 + Hn2 + Hn3 - Hn
n1 n2 n3      /     \
            /         \
           s            g         Jg = Hg1 + Hg2 + Hg3 - Hg
         / | \        / | \       Js = Hs1 + Hs2 + Hs3 - Hs
       s1 s2 s3     g1 g2 g3  -----------------------------
                                Jtot =   Hs1 + Hs2 + Hs3
                                       + Hg1 + Hg2 + Hg3
                                       + Hn1 + Hn2 + Hn3
                                       - Hn,s,g
```

Figure 22: Multiplicative polychotomic tree representing the derivation of nationality (n), sex (s), and generation (g) in two steps. First nationality is separated from a combined sex/generation feature, which is in turn separated. Finally, each feature is separated into its components. To the right of each internal node is the interdependence expression for its children. When calculating the sum of the interdependences, all terms cancel except for the entropy of the root and leaves.

In one such scheme, at each stage, we find one "leaf" feature, and one "leftover" partition. We try to maximize the cross-component interdependence of the former, and minimize the cross-feature interdependence between the leaf and leftover. An even simpler one, which only imposes one constraint at a time, is to first find a feature which maximizes the interdependence. Once this is fixed, a leftover partition is found which minimizes its interdependence with the feature. With this algorithm the mean value is -2.03.

If there are q individuals, and the feature is k-ary, the leftover partition is q/k-ary. Usually k is small, and often it is binary. For instance if the feature is nationality, $k = 2$, and the leftover partition is one in which corresponding Englishmen and Italians are paired. A natural way to think of this partition is as a mapping: After one feature is found for the original problem, a mapping is found for reducing the problem to one of half the size. Since any further features must be orthogonal to the first, no effort need be spent on enforcing non-redundancy.[13]

C Comparison of Evaluation Functions

Section 3.4 described the encoding scheme used by MDL/OC and derived an expression for the total message length under that scheme. This appendix describes some alternative schemes that were tried and rejected. Figure 23 compares them all for various solutions to the family relations problem. The second column (Transmit Whole Tuple, :1-disambig, :all-codes) is MDL/OC. The last column (Non-MDL, :interdependence) is the algorithm described in appendix B. It is unfortunate that the value of the evaluation functions is not directly very meaningful. One advantage of evaluating solutions by performance on a test set is that the numbers have some intuitive meaning. Figure 23 therefore gives completion percentages over all $nl = 336$ possible completions from the training set. These percentages are based on the predication algorithm described in section 3.8.

Each (orthogonal!) variation referred to in the table is explained below:

[13] Actually this is not true—there can be no interdependency among different features for the same component, but there can be interdependency in general. Still it has proved to be a very good heuristic to restrict the system to features which are component-wise orthogonal because it greatly restricts the search space, and does not seem to remove desired solutions from it.

Transmit Whole Tuple versus Completion Of the MDL algorithms in the table, the principal distinction is between those which try to optimize the transmission of information about the training tuple as a whole (Transmit Whole Tuple), and those that optimize the transmission of conditional information about one component of the training tuple, assuming the receiver already knows the other two. The latter group uniformly produced less intuitive rankings of the solutions. There are two variations of the completion-based algorithms: whether the partial tuples are only missing the PERSON2 component (:Person2-only), in which case there are 112 completions, or whether all components can be missing (:1-disambig, :3-disambig, Test Set Completion Percentage, and Training Set Completion Percentage), in which case there are 336 completions. Hinton only completed to PERSON2, and the supervised soybean task is similar in that only the disease is ever queried.

:1-disambig versus :3-disambig The theory can either include the relative frequencies of each individual in the whole training set (:1-disambig and :Person2-only), or there can be a separate count for each tuple (:3-disambig and the completion percentage columns).

:used-codes versus :all-codes If a significant fraction of the possible feature-tuples (such as Female-Male-Male) actually occur in the training set, it is more efficient to transmit a code for each explicitly:

```
<length of code for feature-tuple 1>   <code for feature-tuple 1>
...
<length of code for feature-tuple 2^ci>  <code for feature-tuple 2^ci>
```

But if only a small number of these 2^{c_i} codes are used, it is more efficient to transmit fewer but longer items:

```
<first used feature-tuple>  <code length>  <code>
...
<last used feature-tuple>  <code length>  <code>
```

Even when many of the 2^{c_i} codes are used, the gain from using the first coding scheme is small. In other cases, the first scheme can blow up

> exponentially, so from the functional level, the second scheme appears to be a much better bet. However the search space is less smooth near local optima. Consider a good solution in which the features reflect significant regularities in the training tuples. Then a relatively small number of feature-tuples will occur. But a neighboring solution, in which a single individual's feature-vector is changed, may suddenly have many more feature-tuples occur, although each of these newly-introduced ones will only occur once. Thus there are very steep sides to the local optima, and a simulated annealing search algorithm gets stuck easily. Perhaps a better encoding could remedy this problem by having an escape mechanism so that rarely-occuring feature-tuples do not have to have their own code, and instead the training tuples which generate these rare patterns can be transmitted explicitly.

Besides the conceptual failure to address the desirable characteristics of simplicity and non-redundancy, the interdependence evaluation function has the empirical shortcoming of ranking IDENTITY best of all. The completion-based evaluation functions were rejected because they rank BAD best of all. The Transmit Whole Tuple evaluation functions are largely in agreement with their rankings. Note that the feature PERSON does largely the same job as having separate disambiguation codes for each component, because only people occur in the PERSON1 and PERSON2 components, and only relations occur in the RELATION component. So the top half of the solutions are preferred by :3-disambig evaluation functions and the bottom half are preferred by the :1-disambig evaluation functions.

Imagining a theory-language containing a parameter which switched in one of these models, where each model has the same *a priori* probability, it would add a constant number of bits to all solutions. So an MDL approach would advocate using the model whose best solution had the shortest theory. The smallest number in the Transmit Whole Tuple section of the table is 1245, for the single disambiguation, used codes variation. This, then, seems to be the best evaluation function, and the preference ordering in this (first) column to be the best. Hence the solutions in the table are listed from best to worst according to this evaluation function (separating out the solutions which include the feature PERSON from solutions which don't). However pragmatically the search space is somewhat smoother around local minima for the :all-codes functions, for reasons described above, so the evaluation function in the second column was actually chosen for MDL/OC. I'm a little uneasy about the :all-codes functions, becuase they blow up for the IDENTITY feature due to the huge number of possible feature tuples. Also, the best solution for the other two Transmit Whole Tuple evaluation functions are not too much worse, at 1281 and 1287. Although I have wavered about whether it is preferable to make PERSON an explicit feature, my current opinion is that it is best to make everything possible explicit, and to continue to use the :1-disambig variations.

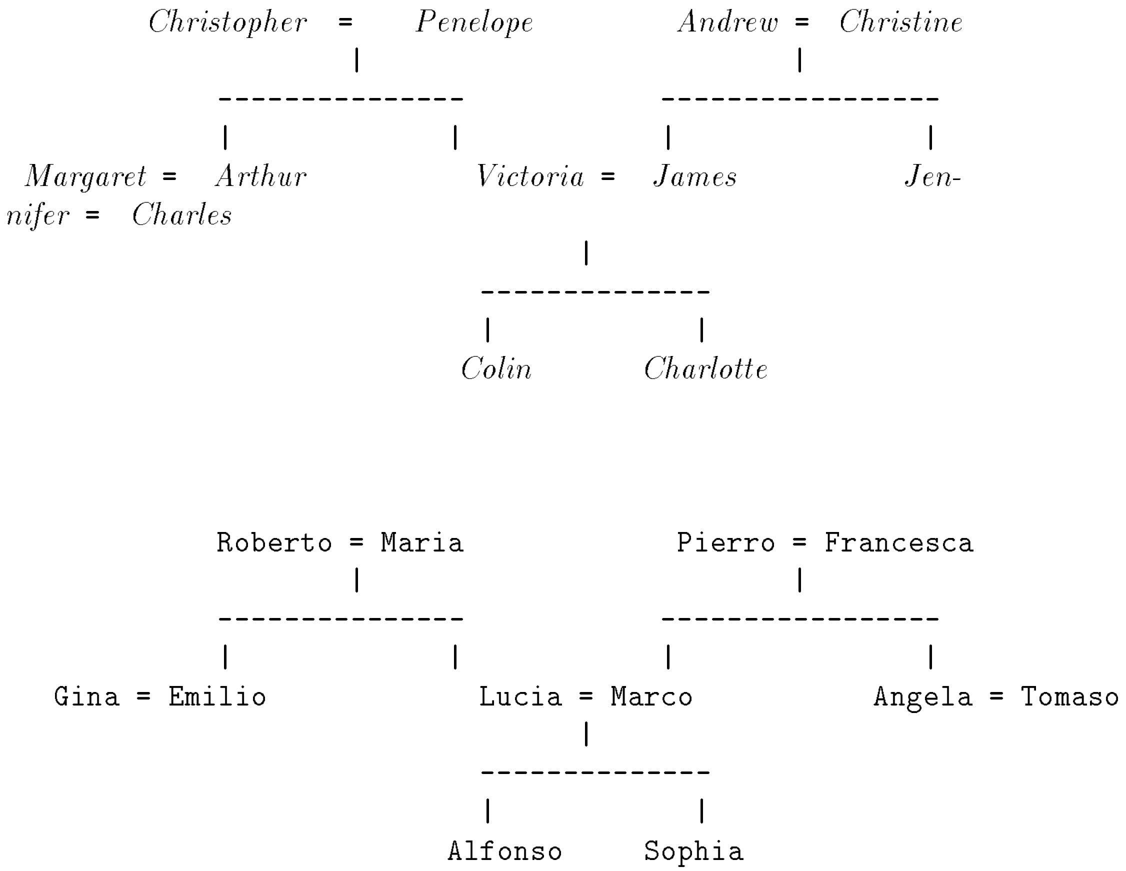
	Transmit Whole Tuple				Completion	
Feature Set	:1-disambig :used-codes	:1-disambig :all-codes	:3-disambig :used-codes	:3-disambig :all-codes	Test Set Completion Percentage	Training Set Completion Percentage
(S N G)	1451	1456	1281	1287	47.0%	54.8%
(S N G B)	1531	1503	1345	1319	57.7%	65.5%
(N G)	1548	1558	1451	1460	23.8%	34.5%
(S N B)	**1577**	**1539**	**1464**	**1428**	**20.8%**	**35.7%**
(S N)	1609	1603	1541	1535	12.5%	22.0%
(N G B)	1627	1606	1515	1492	28.0%	42.3%
(S G)	1637	1643	1424	1430	30.4%	41.1%
BAD	**1652**	**1946**	**1571**	**1865**	**66.1%**	**82.1%**
(N B)	1674	1642	1635	1601	10.1%	22.6%
(S G B)	1702	1676	1487	1461	37.5%	49.4%
(N)	1705	1705	1711	1710	7.1%	14.3%
(G)	**1736**	**1746**	**1594**	**1604**	**15.5%**	**23.8%**
(S B)	1757	1721	1607	1571	14.9%	26.2%
(BAD-F)	1764	2046	1568	1850	1.9%	34.5%
(S)	1796	1792	1684	1679	9.5%	14.9%
(G B)	**1800**	**1779**	**1656**	**1636**	**17.9%**	**29.8%**
(B)	1854	1822	1777	1746	7.1%	16.7%
()	1894	1894	1852	1852	4.8%	9.5%
(RANDOM)	1942	1919	1829	1805	3.0%	10.1%
(IDENTITY)	**3454**	**318557**	**3454**	**318557**	**0.0%**	**94.0%**
(P S N G)	1245	1248	1319	1320	47.0%	54.8%
(P S G)	1340	1343	1461	1464	30.4%	41.1%
(P N G)	1343	1350	1487	1495	23.8%	34.5%
(P S N G B)	**1365**	**1334**	**1382**	**1352**	**57.7%**	**65.5%**
(P S N)	1398	1389	1578	1570	12.5%	22.0%
(P S N B)	1399	1359	1502	1461	20.8%	35.7%
(P G)	1438	1446	1630	1637	15.5%	23.8%
(P S G B)	**1459**	**1430**	**1523**	**1495**	**37.5%**	**49.4%**
(P N G B)	1463	1437	1551	1526	28.0%	42.3%
(P S)	1492	1485	1720	1712	9.5%	14.9%
(P N)	1495	1492	1747	1744	7.1%	14.3%
(P S B)	**1495**	**1454**	**1645**	**1604**	**14.9%**	**26.2%**
(P N B)	1498	1461	1671	1636	10.1%	22.6%
(P G B)	1557	1532	1694	1669	17.9%	29.8%
(P)	1590	1587	1890	1887	4.8%	9.5%
(P B)	**1591**	**1557**	**1813**	**1779**	**7.1%**	**16.7%**

Figure 23: Each row in the table lists the value of the evaluation functions for a particular solution. These solutions are sets of the following partitions: (S) SEX, (N) Nationality, (G) Generation, (B) Branch of the Family, (P) PERSON, (RANDOM) a random even binary partition, (IDENTITY) a partition in which each individual is assigned a distinct element, BAD a non-intuitive solution found by using a completion-based MDL evaluation function, (BAD-F) one of the partitions in BAD. Figures 24-33 diagram these features. It is interesting that the acceptable evaluation functions in figure 23 are nearly unanimous in preferring the solution SEX PERSON NATIONALITY GENERATION and rejecting a BRANCH OF FAMILY feature.

Completion						Non-MDL
:3-disambig :used-codes	:3-disambig :all-codes	:Person2-only :used-codes	:Person2-only :all-codes	:1-disambig :used-codes	:1-disambig :all-codes	:interdependence
737	688	329	280	906	858	-2.56
691	594	374	286	876	779	-2.11
1020	982	428	391	1118	1081	-1.98
1128	**1063**	**443**	**387**	**1239**	**1174**	**-1.81**
1313	1297	457	439	1381	1365	-1.43
974	889	472	397	1088	1001	-1.53
993	951	439	394	1206	1163	-1.86
633	**485**	**452**	**280**	**714**	**566**	**-2.20**
1411	1358	542	498	1451	1397	-1.12
948	856	482	400	1163	1072	-1.41
1597	1591	555	550	1591	1587	-0.69
1277	**1245**	**537**	**505**	**1418**	**1385**	**-1.28**
1385	1326	551	501	1534	1474	-1.12
1130	1060	571	478	1327	1258	-1.93
1570	1558	566	553	1682	1672	-0.73
1231	**1150**	**581**	**511**	**1373**	**1294**	**-0.84**
1668	1619	651	612	1746	1697	-0.43
1852	1852	664	664	1894	1894	0.00
1854	1805	705	672	1966	1917	0.00
2442	**46960**	**2007**	**6873**	**2442**	**46960**	**-3.96**
773	723	366	315	701	649	-2.56
1030	984	475	428	909	864	-1.86
1056	1016	466	426	912	871	-1.98
727	**628**	**410**	**319**	**711**	**612**	**-2.11**
1350	1330	493	473	1169	1150	-1.43
1164	1096	480	421	1062	995	-1.81
1313	1278	574	540	1121	1086	-1.28
984	**890**	**518**	**433**	**919**	**825**	**-1.41**
1011	922	509	430	922	834	-1.53
1606	1593	602	587	1378	1365	-0.73
1633	1624	593	584	1381	1372	-0.69
1421	**1359**	**589**	**535**	**1271**	**1209**	**-1.12**
1448	1391	579	532	1274	1218	-1.12
1267	1184	617	544	1131	1047	-0.84
1890	1887	701	698	1590	1587	0.00
1704	**1653**	**688**	**646**	**1482**	**1431**	**-0.43**

D Desired Features for Family Relations Problem

These figures define the features referred to in the paper as NATIONALITY, SEX, GENERATION, 2-WAY GENERATION, SKEWED 2-WAY GENERATION, BRANCH OF FAMILY, PERSON, PARENT, RANDOM, and BAD-F. The features as shown are not canonical, due to symmetries. For instance in the NATIONALITY feature, the RELATION component is uninformative. Hence it is unimportant whether relations are grouped with the English or the Italians. The key table for each figure indicates which font corresponds to which named partition, and the single-letter abbreviation for it used in the main text of the paper. The names for the partitions, such as "Italian," are assigned by the experimenter for ease of explanation; it is an important future task to partially automate the process of assigning intensional descriptions to the partitions.



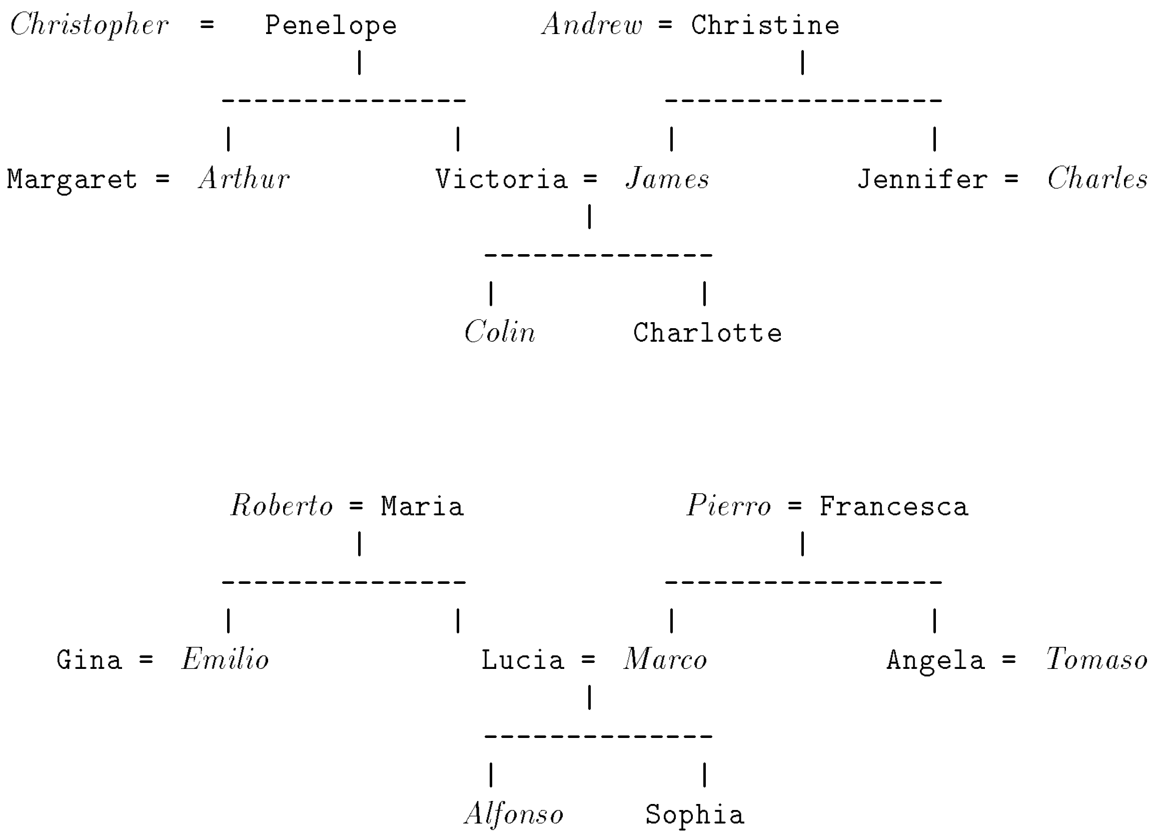
brother	husband
sister	wife
nephew	son
niece	daughter
uncle	father
aunt	mother

Key

E English

I Italian

Figure 24: The nationality feature.



brother	*husband*
sister	wife
nephew	*son*
niece	daughter
uncle	*father*
aunt	mother

Key
M Male
F Female

Figure 25: The sex feature.

```
CHRISTOPHER  =   PENELOPE        ANDREW = CHRISTINE
                     |                          |
             ---------------            -----------------
             |             |            |               |
Margaret = Arthur         Victoria = James         Jennifer = Charles
                                   |
                             --------------
                             |            |
                            Colin      Charlotte
```

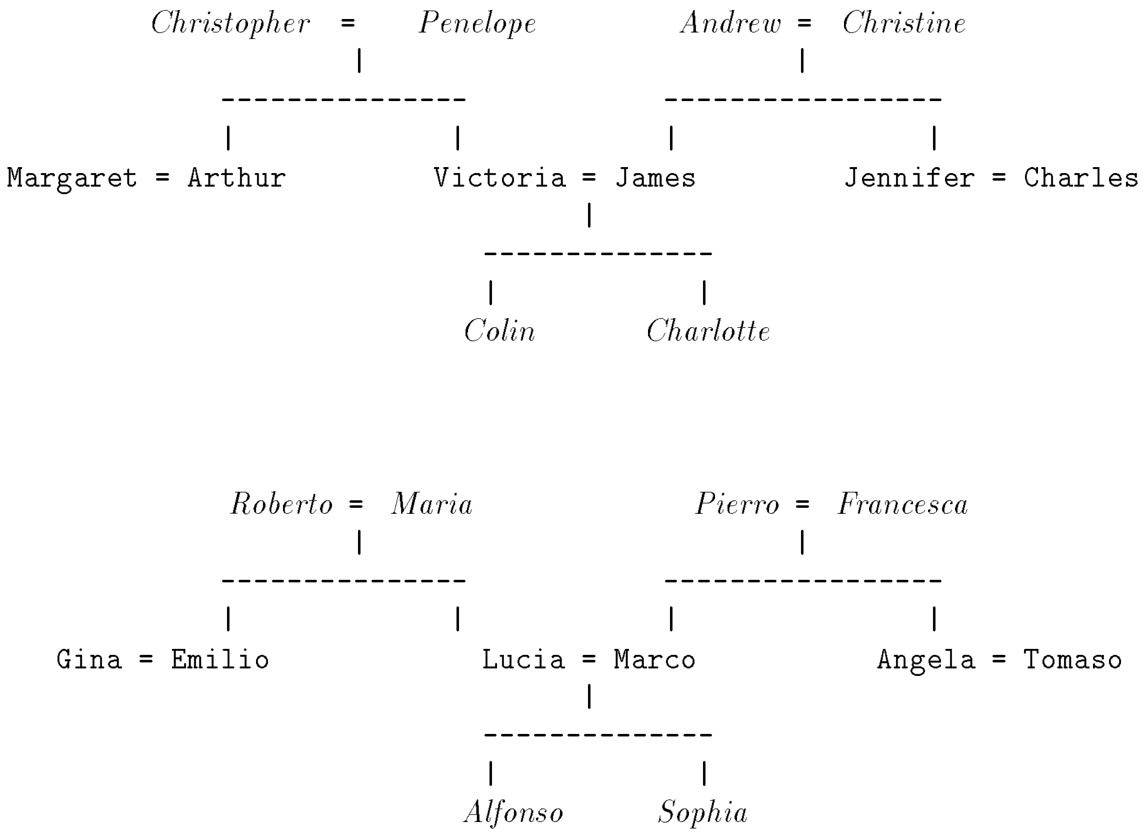
```
             ROBERTO = MARIA             PIERRO = FRANCESCA
                     |                          |
             ---------------            -----------------
             |             |            |               |
   Gina = Emilio             Lucia = Marco           Angela = Tomaso
                                   |
                             --------------
                             |            |
                            Alfonso    Sophia
```

brother	husband
sister	wife
nephew	*son*
niece	*daughter*
UNCLE	FATHER
AUNT	MOTHER

Key

3 3rd-Generation
2 2nd-Generation
1 1ST-GENERATION

Figure 26: The generation feature.



brother	*husband*
sister	*wife*
nephew	son
niece	daughter
uncle	father
aunt	mother

Figure 27: The 2-way-generation feature.

```
          Christopher  =    Penelope           Andrew =  Christine
                       |                                |
                 ---------------          -----------------
                 |             |          |               |
Margaret = Arthur           Victoria = James          Jennifer = Charles
                                     |
                               --------------
                               |            |
                             Colin       Charlotte


                 Roberto = Maria               Pierro = Francesca
                       |                                |
                 ---------------          -----------------
                 |             |          |               |
 Gina =   Emilio                 Lucia =   Marco               An-
gela =  Tomaso
                                     |
                               --------------
                               |            |
                            Alfonso     Sophia
```

brother	*husband*
sister	*wife*
nephew	son
niece	daughter
uncle	father
aunt	mother

Figure 28: The skewed-2-way-generation feature.

```
Christopher  =   Penelope        Andrew = Christine
                     |                          |
             ---------------            -----------------
             |             |            |               |
MARGARET =   Arthur         Victoria = James           Jennifer = CHARLES
                                   |
                             --------------
                             |            |
                           Colin      Charlotte
```

```
             Roberto = Maria             Pierro = Francesca
                     |                          |
             ---------------            -----------------
             |             |            |               |
   GINA =   Emilio            Lucia = Marco            Angela = TOMASO
                                   |
                             --------------
                             |            |
                           Alfonso     Sophia
```

brother	husband
sister	wife
NEPHEW	son
NIECE	daughter
UNCLE	father
AUNT	mother

Key

I Intermediate

C Central

O OUTSIDE

Figure 29: The "Branch of Family" feature.

```
          Christopher  =     Penelope            Andrew =  Christine
                       |                                 |
                 ---------------               -----------------
                 |             |               |                |
 Margaret =   Arthur            Victoria =   James              Jen-
nifer =   Charles
                                     |
                               --------------
                               |            |
                             Colin       Charlotte


               Roberto =  Maria                 Pierro =  Francesca
                       |                                 |
                 ---------------               -----------------
                 |             |               |                |
   Gina =     Emilio              Lucia =    Marco              An-
gela =   Tomaso
                                     |
                               --------------
                               |            |
                            Alfonso      Sophia
```

brother
sister
nephew
niece
uncle
aunt

husband
wife
son
daughter
father
mother

Key

P Person
R Relation

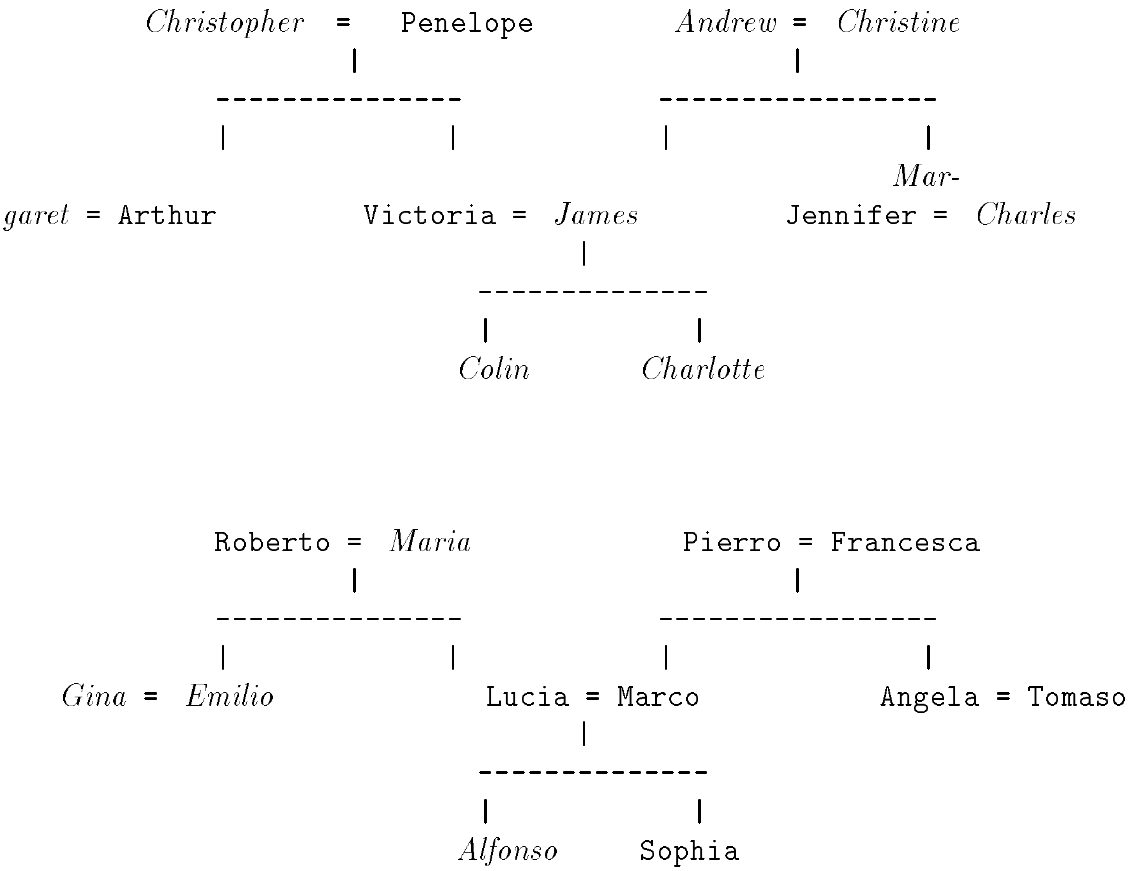
Figure 30: The PERSON feature.

```
         Christopher =   Penelope       Andrew =  Christine
                     |                          |
             ---------------           -----------------
             |             |           |               |
Margaret = Arthur                                    Victo-
ria =  James        Jennifer = Charles
                                   |
                             --------------
                             |            |
                           Colin      Charlotte
```

```
              Roberto =  Maria                 Pierro =  Francesca
                     |                          |
             ---------------           -----------------
             |             |           |               |
   Gina = Emilio                                       Lu-
cia =  Marco          Angela = Tomaso
                                   |
                             --------------
                             |            |
                           Alfonso     Sophia
```

brother	husband
sister	wife
nephew	*son*
niece	*daughter*
uncle	*father*
aunt	*mother*

Figure 31: The Parent feature.



brother	*husband*
sister	wife
nephew	son
niece	*daughter*
uncle	*father*
aunt	mother

Figure 32: The RANDOM feature.

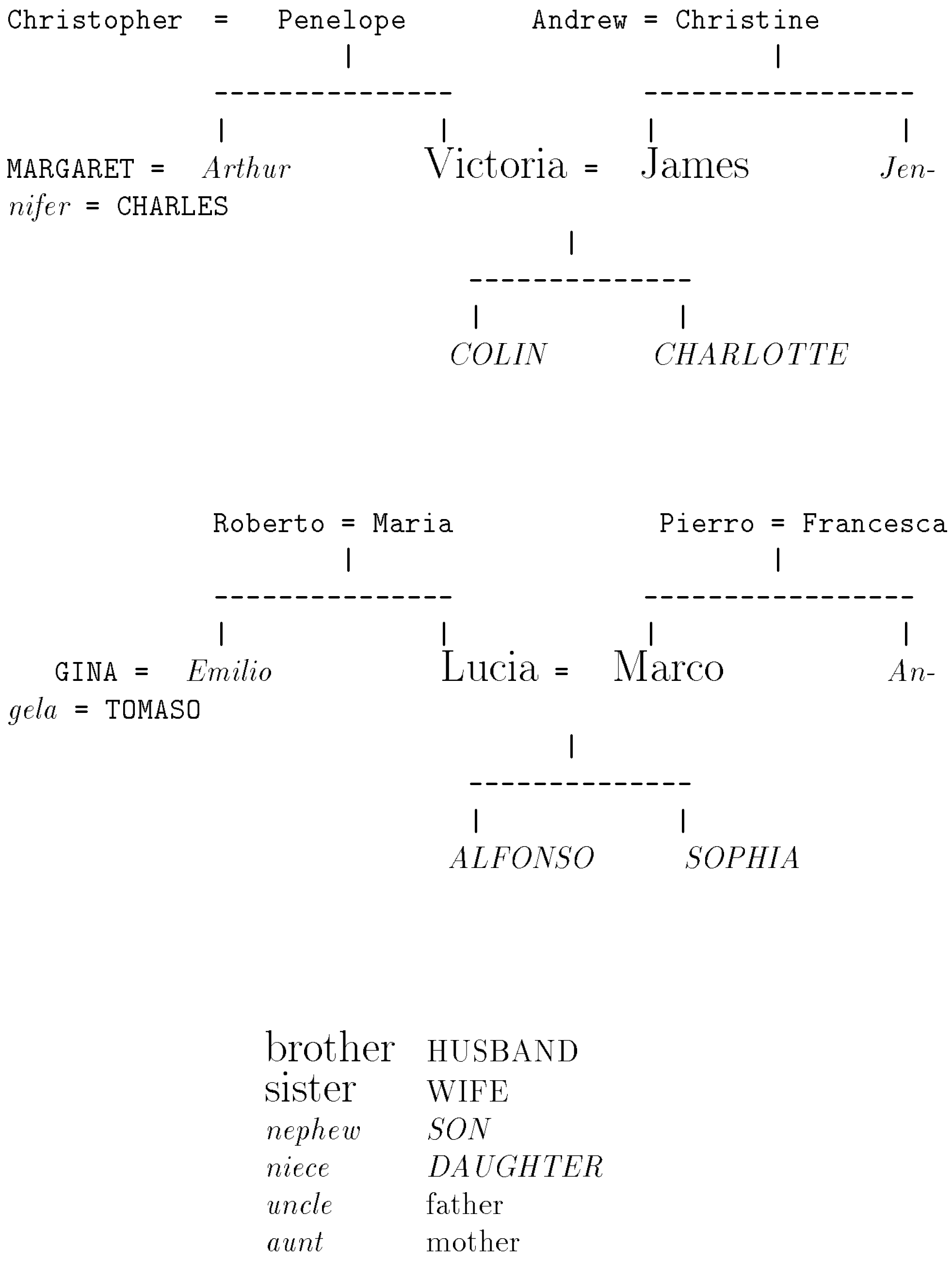


Figure 33: The unintuitive quinary feature, which encodes both generation and branch of family, found with the completion evaluation functions, BAD-F.

References

[Becker and Hinton, 1989] Suzanna Becker and Geoffrey E. Hinton. Spatial coherence as an internal teacher for a neural network. Technical Report CRG-TR-89-7, University of Toronto, December 1989.

[Chaitin, 1977] G. J. Chaitin. Algorithmic information theory. *IBM Journal of Research and Development*, pages 350–359, July 1977.

[Cheeseman *et al.*, 1988] Peter Cheeseman, Don Freeman, James Kelly, Matthew Self, John Stutz, and Will Taylor. Autoclass: A Bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 54–64, 1988.

[Chernoff and Moses, 1959] Herman Chernoff and Lincoln E. Moses. *Elementary Decision Theory.* Wiley, New York, 1959.

[Duda and Hart, 1973] Richard Duda and Peter Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, New York, 1973.

[Falkenhainer *et al.*, 1989] Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. The structure-mapping engine: Algorithms and examples. *Artificial Intelligence*, 41:1–63, 1989.

[Fisher and Schlimmer, 1988] Doug Fisher and Jeff Schlimmer. Concept simplification and predictive accuracy. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 22–28, Ann Arbor, Michigan, 1988.

[Galland and Hinton, 1990] Conrad C. Galland and Geoffrey E. Hinton. Experiments on discovering high order features with mean field modules. Technical Report CRG-TR-90-3, University of Toronto, February 1990.

[Gao and Li, 1989] Qiong Gao and Ming Li. The minimum description length principle and its application to online learning of handprinted characters. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 843–848, Detroit, Michigan, 1989.

[Gentner, 1983] Dedre Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2), 1983.

[Greiner, 1988] Russell Greiner. Learning by understanding analogies. *Artificial Intelligence*, 35:81–125, 1988.

[Hinton and Sejnowski, 1986] Geoffrey E. Hinton and Terrence J. Sejnowski. Learning and relearning in Boltzmann Machines. In David E. Rumelhart, James L. McClelland, and the PDP research group, editors, *Parallel distributed processing: Explorations in the microstructure of cognition. Volume I*, chapter 7, pages 282–317. Bradford Books, Cambridge, MA, 1986.

[Hinton, 1986] Geoffrey E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Cognitive Science Conference*, pages 1–12, Amherst, Massachusetts, 1986. Cognitive Science Society.

[Lenat and Guha, 1990] Douglas B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems.* Addison-Wesley, Reading, MA, 1990.

[Lucassen, 1983] John M. Lucassen. Discovering phonemic base forms automatically: An information theoretic approach. Technical Report RC 9833 (# 43527), IBM, February 1983.

[Michalski and Chilausky, 1980] Ryszard S. Michalski and R. L. Chilausky. Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems*, 4(2), 1980.

[Michalski, 1983] Ryszard S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20:111–161, 1983.

[Michie and Al-Attar, 1990] Donald Michie and A. Al-Attar. Use of sequential Bayes with class probability trees. In J. E. Hayes, D. Michie, and E. Tyugu, editors, *Machine Intelligence 12.* Oxford University Press, Oxford, 1990.

[Pagallo and Haussler, 1990] Guilia Pagallo and David Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–99, 1990.

[Peterson and Anderson, 1987] Carsten Peterson and James R. Anderson. A mean field theory learning algorithm for neural networks. Technical Report MCC-EI-259-87, Microelectronics and Computer Technology Corporation, Austin, Texas, 1987.

[Peterson and Soderberg, 1989] Carsten Peterson and Bo Soderberg. A new method for mapping optimization problems onto neural networks. Technical Report LU-TP-89-1, University of Lund, March 1989.

[Quinlan, 1987] J. Ross Quinlan. Generating production rules from decision trees. In *Proceedings of the Tenth International Joint Conference on*

Artificial Intelligence, pages 304–307, Milan, 1987.

[Rissanen, 1978] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[Tan and Eshelman, 1988] Ming Tan and Larry Eshelman. Using weighted networks to represent classification knowledge in noisy domains. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 121–134, Ann Arbor, Michigan, 1988.

[Tversky, 1977] Amos Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.

[Watanabe, 1969] Satosi Watanabe. *Knowing and Guessing.* John Wiley and Sons, New York, 1969.