A Minimal Encoding Approach to Feature Discovery

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

Abstract

This paper discusses unsupervised learning of orthogonal concepts on relational data. Relational predicates, while formally equivalent to the features of the concept-learning literature, are not a good basis for defining concepts. Hence the current task demands a much larger search space than traditional concept learning algorithms, the sort of space explored by connectionist algorithms. However the intended application, using the discovered concepts in the Cyc knowledge base, requires that the concepts be interpretable by a human, an ability not yet realized with connectionist algorithms. Interpretability is aided by including a characterization of simplicity in the evaluation function. For Hinton's Family Relations data, we do find cleaner, more intuitive features. Yet when the solutions are not known in advance, the difficulty of interpreting even features meeting the simplicity criteria calls into question the usefulness of any reformulation algorithm that creates radically new primitives in a knowledge-based setting. At the very least, much more sophisticated explanation tools are needed.

Introduction

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 that can discover some of this missing information would be helpful, but it is crucial to be able to explain new concepts in order to properly integrate the discovered knowledge.

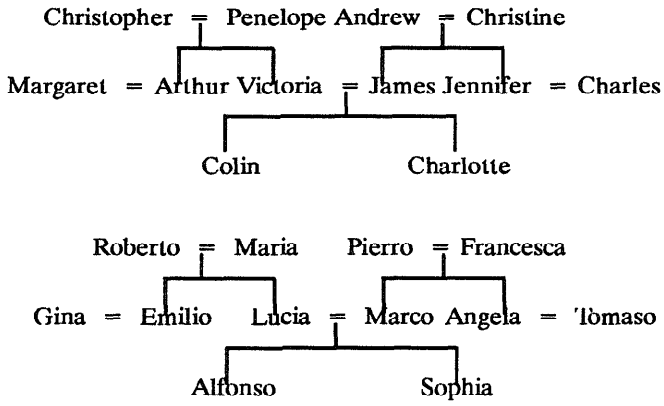
In this paper, concept learning is cast as assigning *features* to individuals based on training examples consisting of tuples of those individuals. Each possible value of each feature represents a concept. For instance, the feature COLOR generates the concepts Red, Blue, etc. Hopefully such an algorithm 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 Family Relations problem is described in the next section. Then the Minimum Description Length (MDL) principle [Rissanen, 1989], in which theories are judged to be good in direct proportion to how small they are, is very briefly described followed by its particular use in feature discovery on the family relations problem. Finally, some implications of this work on the general problem of reformulation are discussed in light of experience on real Cyc data. The algorithm is called MDL/OC, for Minimum Description Length/Orthogonal Clustering. A more detailed description, including results on a diagnosis problem also used by Michalski and Chilausky [1980], real Cyc data, and an analogical mapping also used by Falkenhainer *et al* [1989], is given in Derthick [1990].

Problem Representation

For the purposes of this paper, the goal of a learning knowledge representation system is to develop a domain theory from a set of ground atomic assertions that allows it to decide the truth or falsity of other ground assertions. The Family Relations problem's assertions use only binary predicates, which are usually called *features* in the concept learning literature. MDL/OC can handle discrete predicates of arbitrary arity.

The family relations training data consists of the 112 3-tuples representing true family relationships in the family trees shown in Figure 1(top). The elements of the tuples include the 24 people and the 12 predicates husband, wife, son, daughter, father, mother, brother, sister, nephew, niece, uncle, aunt. These predicates make for poor features; the set of people that have the value Pierro for the feature father, for instance, has only two members, so it is not very good for generalizing over. A decision tree algorithm that does not redescribe the input by constructing new terms would construct a shallow tree that has a large branching factor. And previous constructive induction algorithms, which only consider simple boolean combinations of existing features or values, would not do much better. Another way to see the poor match of this data to previous concept learning algorithms is to look at the feature-vector representation of the problem in Figure 1(bottom). There



(Christopher Wife Penelope)
(Christopher Son Arthur)
(Christopher Daughter Victoria)

```
                HWSDFMBSNN  U  A
Christopher : -PAV------  -  -
Charlotte   : ----JVC---{CA}{JM}
Arthur      : -M--CP-VCC  -  -
```

Figure 1: Top, the family trees from which the 112 training tuples are derived. "=" means "spouse," and lines mean "child." Middle, a few examples of the training tuples for the family relations problem, in the syntax accepted by MDL/OC. It is possible to use an feature-value representation of this domain (Bottom), in which the given predicates Husband, Wife, Son, Daughter, Father, Mother, Brother, Sister, Nephew, Niece, Uncle, Aunt induce an feature vector on each individual.

are many null values and some multi-values. The number of values of each feature is large.

Much more desirable would be to construct new predicates that make for more natural features, such as Sex, Nationality, or Generation. These features would still take a person as an argument, but their values would be constructed: Male, Female, English, Italian, Old, Middle, and Young. Of course the algorithm cannot come up with English names for its features and values. But this paper would be even less readable using gensyms, so it is important to remember that only the 24 people and 12 *given predicates* appear in the input, and that other features and values are names I attached to partitions and their elements constructed by the algorithm. Somewhat confusingly, MDL/OC does not distinguish between predicates and arguments. It interprets the training set strictly as a set of tuples, and discovers new ways to categorize the elements. To emphasize this, the given predicates and people are sometimes collectively called *individuals*. The feature that distinguishes them is useful to the algorithm, so Type is another constructed feature, with values Person and Given-Predicate.

In algorithms that represent an individual by its feature-vector, all assertions about one individual are present in a single training example, and testing is accomplished by completing one or a few missing features values of a new individual. In MDL/OC, a training example consists of only a single assertion, of the form (PERSON1 GIVEN-PREDICATE PERSON2). Testing generalization on new ground assertions (termed "completion") is done by a separate system. The intent is to use the discovered features to help extend a knowledge base with a subtheory of the training domain, and to have completion done by the knowledge-based system. However a simple-minded algorithm that does not require a human to integrate the features with existing concepts in a KB is described in the evaluation of results section.

MDL Feature Discovery

Coding Scheme

MDL is a very powerful and general approach that can be applied to any inductive learning task involving sufficient data. It appeals to Occam's razor—the intuition that the simplest theory that explains the data is the best one. The simplicity of the theory is judged by its length under some encoding scheme 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. There is a tradeoff between having a large theory that requires few additional data bits to explain the particular training set, and having a small theory that requires much of the information to be left as data. To ensure that the encoding scheme is information-preserving, the task is often thought of as transmitting the training set to a receiver using the fewest bits. There must be an algorithm for the receiver to reconstruct the data from the compressed form.

This subsection assumes that useful features like Sex have already been found, and the concern is only to describe how they are used to encode and decode the training data, and to measure the resulting encoding length. This length serves as a declarative *evaluation function* over sets of features. The next section describes how a weak method is used to search for a set of features that minimizes the evaluation function.

MDL/OC's encoding scheme reflects two characteristics I believe intuitively good features should have: They should enable simple expression of the domain constraints, and they should not be redundant. Redundancy is minimized by any MDL scheme; simplicity is enforced by constraining each feature separately. For instance, the Sex feature of the GIVEN-PREDICATE element of a tuple perfectly predicts the Sex feature of the PERSON2 element, independent of the type, Nationality, Generation, and branch of the family. For instance, Mother has the value Female, and any motherhood assertion will have a Female PERSON2. The other features listed have their own independent regularities. Once each feature of a missing testing tuple element has been found, the individuals with these feature-values can be looked up. MDL/OC gets its

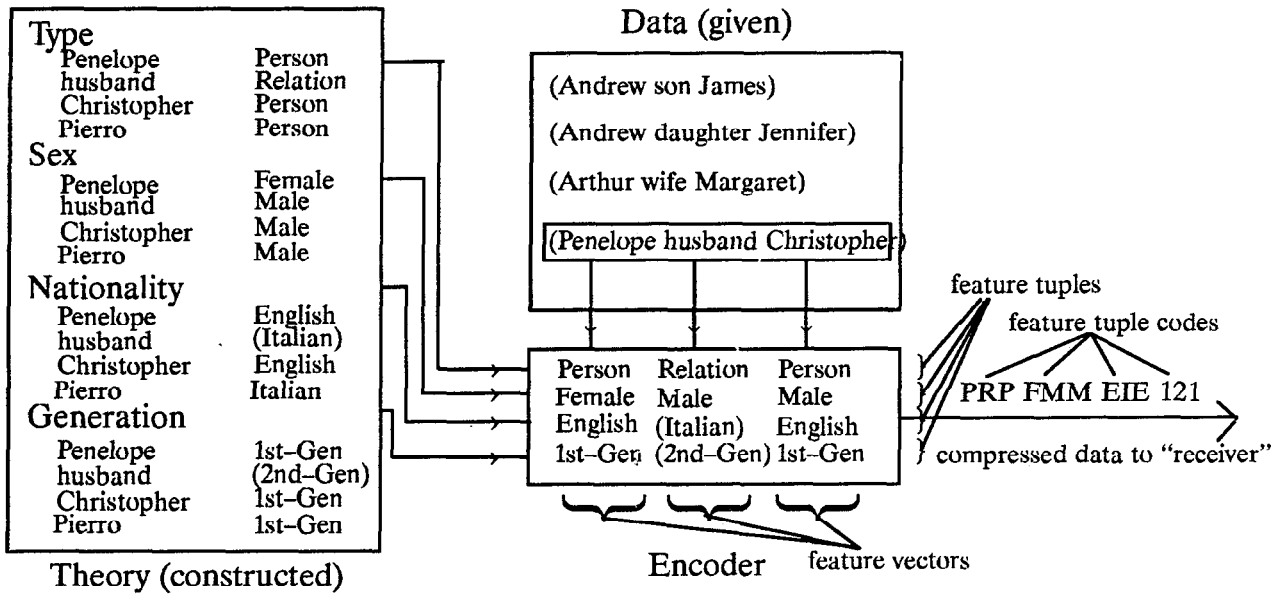


Figure 2: The encoding scheme used by MDL/OC. To transmit the training example (Penelope husband Christopher), the encoder looks up the feature-vectors for each component of the tuple (columns). Each row then represents a feature-tuple to be transmitted, using code-words that have previously been sent to the receiver. The feature value names in this figure are attached by the experimenter as an aid to understanding. When applied to a given predicate such as husband, the label "Italian" is in parentheses to indicate that, while husband formally has the same value for this feature as Emilio, it is not really meaningful; no pairs of rules differ only in their treatment of the Nationality feature of the GIVEN-PREDICATE of a tuple. "2nd-Gen" is in parentheses because it really means "same generation" when applied to a GIVEN-PREDICATE. This overloading doesn't cause problems because the subset of the domain that can appear as a GIVEN-PREDICATE is disjoint from that that can appear as a PERSON1 or PERSON2.

power from discovering features that makes this kind of underlying regularity apparent. The convenience afforded by assumptions of independence is embraced almost universally in Bayesian inference. It is surprising that there have been few attempts to discover orthogonal concepts.

One straightforward encoding scheme would assign each individual a code-word and transmit the elements of each training tuple in order. However this scheme ignores the regularities that hold across each training tuple. Since we are looking for regularities across orthogonal features, MDL/OC's coding scheme (Figure 2) transmits tuples of feature-values across individuals (called a *feature-tuple*) for each feature, as opposed to a tuple of feature-values across features (called a *feature-vector*) for each individual. At the receiver, the process is just the reverse. The code-words representing feature-tuples are decoded and rearranged into feature-vectors, which are decoded into individuals. In case multiple individuals have the same feature vector, the message is augmented with disambiguation code-words. For instance, using the features in Figure 2, both Penelope and Christine have the feature-vector PFE1, so every training tuple in which one of them appears will require an extra bit.

Using the notation in figure 3, what ends up being transmitted for each training tuple, therefore, is d (= 4 in the figure) feature tuples, rather than n (= 3 for Family Relations) feature vectors. This is a win if the codes for feature tuples are shorter than codes for feature vectors by at least n/d, on average. For Family Relations, the average code length for feature vectors is $H(f(\mathcal{T})) = 5.1$ bits. The average feature tuple code lengths, $H(f_i(\mathcal{S}))$, are: Type = 0.0, Sex = 1.9, Nationality = 1.0, Generation = 2.6. So this domain has regularities allowing feature tuple codes that are indeed much shorter than feature vector codes.

By Shannon's coding theorem, and assuming optimal codes, the number of bits required to transmit each symbol is the negative log of the symbol's probability of occurrence. Summing this for all the feature-tuple and disambiguation codewords gives the total number of bits for the data term. (Any real code will require an integral number of bits. However, all the codes in this paper are "simulated," and the lengths are calculated using information-theoretic formulas whose range is the non-negative real numbers.)

The theory description consists of the number of features (d), individuals (q), training examples (l), training tuple size (n), number of values for each feature (c_i), the assignment of individuals to feature vectors (f), and the feature-tuple and disambiguation codes. There is insufficient space here to derive the evaluation function in any detail. In principle, it is just turning the crank. A code table can be transmitted as a histogram of symbol frequencies; both transmitter and receiver can apply

Variable	Meaning
$q = 36$	Domain size
$l = 112$	Training Set size
$n = 3$	Training Tuple Arity
$d = 5$	Feature Vector size
$c_i = 2$	Feature arities
$\mathcal{S}$	Random variable ranging over training tuples. Uniform probabilities.
$\mathcal{T}$	Random variable ranging over individuals. Probabilities match the training data.
$f(\mathcal{T})$	Feature vector for individual $\mathcal{T}$
$f_i(\mathcal{T})$	ith feature value for individual $\mathcal{T}$.
$f_i(\mathcal{S})$	ith feature tuple for training example $\mathcal{S}$.
$H(\cdot)$	Entropy of a random variable = the expected number of bits to encode a value of the variable = $\sum \Pr(\cdot) \log \Pr(\cdot)$

Figure 3: Notation used in specifying the MDL/OC evaluation function. Variable values are given for the Family Relations problem and for the search parameters given in the search algorithm section.

a previously agreed-on algorithm to derive an optimal code from this. Since the total number of symbols (nl) and the alphabet size ($\sum c_i^n$) are known to the receiver, straightforward counting arguments lead to a formula for the number of bits to transmit the code tables.

In practice, however, this efficient but abstract approach does not produce a smooth evaluation function. Although it takes about twice as many bits, the search works much better with an evaluation function based on transmitting the code tables entry by entry. Each entry specifies the code for one symbol. Quinlan (personal communication) found the same factor of two worsening in total number of bits, along with improved search performance, when he used a less abstract coding scheme in his MDL approach to learning decision trees [Quinlan and Rivest, 1989]. Here this effect is apparently due to smoothness, because the intuitive feature sets are local optima in both models.

Several approximations are made to further enhance the smoothness of the evaluation function. The average number of bits to specify the codes in the code tables is approximated by the entropy of the random variable being coded (or the conditional entropy of the random variable given the feature-vector in the case of the disambiguation code). This corresponds to weighting the average using the observed probabilities in the data. The number of bits to transmit an integer z is approximated $\log z$. Finally, 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. I found no smooth approximation for d, which is expected to be very small, so it is simply dropped.

Features	Theory	Feature Tuples	Disambiguation	Total
(T S N G)	298	625	339	1262
(T S N G B)	349	922	78	1348
(T G)	265	296	899	1460
(T S)	215	218	1066	1500
(T N)	215	113	1179	1507
(T B)	238	296	1036	1571
(T)	199	0	1402	1601
()	197	0	1711	1908
(RANDOM)	222	335	1375	1933
(IDENTITY)	317,808	762	0	318,570

Table 1: The value of the evaluation function for several sets of features that I hand-constructed to solve the problem. The units are bits. T=Type, S=Sex, N=Nationality, and RANDOM are 2-valued; G=Generation and B=Branch are 3-valued; IDENTITY is 36-valued. When constrained to find five binary features, MDL/OC achieved a slightly better score (1260) using a 2-valued version of Generation, and a feature distinguishing the parents from the non-parents instead of Branch.

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

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

The constant terms, $H(\mathcal{T})$ and $\log qln$, can be ignored by the optimization algorithm. If there are no features, f contains no information, and this reduces to $(nl+q) \cdot H(\mathcal{T}) + \log qln$. This is a more precise calculation for the straightforward encoding scheme mentioned above in which each tuple element is transmitted separately. As f becomes more informative, $H(\mathcal{T})$ is reduced by $H(f(\mathcal{T}))$, but the cost of encoding f increases. In the other limit, where everything is included in the theory, there is a single feature with $q = 36$ values, one for each individual. The feature-tuple code table will contain $q^n = 46,656$ entries, so this evaluation function, based on transmitting the table explicitly, blows up (Table 1).

Search Algorithm

By virtue of the above declarative specification of what constitutes a good set of features, the learning task has been cast as a well-defined optimization problem: minimize $E(f)$ over all possible sets of partitions. Initially the individuals are assigned random values for each feature. This constitutes the initial feature-set hypothesis. The search procedure thus requires the maximum number of features sought, d_{max}, and the maximum number of values for each, $c_{i_{max}}$, to be fixed in advance.

The search is then carried out by simulated annealing. This is a generalization of hill-climbing, and like

hill-climbing the search proceeds by evaluating the evaluation function for a configuration of the search space that neighbors the currently hypothesized configuration. Depending on this value, we may "move" to the neighboring configuration as the new hypothesis. In MDL/OC, neighboring feature sets in the search space are those for which the value of a single feature of a single individual differs. For instance, a move might consist of changing the value of Pierro's Nationality feature from Italian to English. Since so little is changed between neighboring feature sets, it is expected that the evaluation function will change smoothly. This allows the search to descend gradually and hopefully find the optimal feature set. I have tried more complicated kinds of moves, but this topology has worked best.

Simulated annealing differs from hill climbing in that uphill moves are sometimes made, in an effort to avoid becoming stuck in local optima. The greater the improvement in the evaluation function, the greater the chances of accepting a move. The amount of randomness in the decision is controlled by the *temperature*, T, which is gradually decreased during search. Numerically, $\Pr(move) = \frac{1}{1+e^{-(E_{new}-E_{current})/T}}$.

The parameters used for the Family Relations problem were as follows: Anneal from an initial temperature of 500.0 and gradually decrease until the probability of accepting any move falls 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 .999999. This takes about four hours on a Symbolics 3630.

Before running any experiments, I constructed what I expected would be the best set of features for this problem. Table 1 shows that the evaluation function ranks combinations of these features in a reasonable way. However MDL/OC finds slightly different features. This is acceptable as long as they at least equal the expected ones according to both the evaluation function and intuition. Finding five binary features, the results obtained over 20 trials were as follows: It always found Type, Sex, and Nationality. It found Parental-Status (with two values, Parent and Non-Parent) 19 times, and a skewed version the other time. It found a 2-valued version of Generation (with values Middle-Generation and Old-or-Young-Generation) 13 times, and a skewed version the other 7 times. Thus a total of 92 of 100 discovered features were ones to which intuitive English descriptions can be attached, and the remaining ones were recognizable variants. Annealing ten times as fast, 83% met this criterion; with another factor of 10 reduction in search time the result was 51%. Hill climbing gives only 15%. A greedy algorithm that begins with the identity feature and successively combines the pair of feature values that results in the shortest message length also did very poorly. Noise immunity was quite good, achieving 92% when 10% of the 112 correct patterns were replaced by tuples in which each element was randomly selected from the 36 individuals, and 64% when 25% of the patterns were replaced.

For these features the encoding length is 1260, marginally better than for my solution. In retrospect, Parental-Status is more intuitive than my attempt to define Branch-of-the-Family.

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. The best theoretical bound that can be proven for simulated annealing is proportional to the search space size, which is multiply exponential. In practice, simulated annealing often works reasonably fast, so useful scaling estimates can only be found empirically. But it is hard to compare search time across domains, 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. Using this criterion, the largest problem I have tried has 30 times more training examples, 200 times more individuals, and the same values for d, n, and the c_i. It requires between three and four orders of magnitude more cpu time. Without more experiments, all that can be said is that it is worse than linear and much better than exponential. One way to fight this scaling is to shrink the search space by finding only a few features and then "freezing" them while more are sought.

Evaluation of Results

Just finding features is of no use unless they can be used to infer new facts, and the ultimate goal of this research is to do this by extending the Cyc knowledge base with new concepts and rules based on the discovered features. But even without understanding the features and extending the KB by hand, it is possible to do completion by considering the feature-tuple frequencies as simple kinds of rules. Note that the following procedure is not part of MDL/OC, but only an attempt render its results in a form suitable for quantitative comparison with other algorithms. For example, if the assertion (Jennifer brother James) had been left out of the training set, the completion for (Jennifer brother ?) could still be found from the five binary features found by MDL/OC as follows: The feature-tuples that occur in the training data for Sex are FMM, MMM, MFF, and FFF. Only FMM matches the value of this feature for the two known elements of the testing tuple, so the missing element must have the value Male. By similar reasoning, the missing element must have Type=Person, Nationality=English, and Generation=Middle. For Parental-Status, Jennifer has the

value NonParent, and brother has the value Parent, and both the patterns NPP and NPN occur in the training data. However the former occurs 19 times (leaving out the current assertion) and the latter only 4. Therefore we guess that the missing element is a Parent. James is the only individual with the guessed feature vector. However using these five features, even when the feature-tuple frequencies are based on all 112 assertions, completion percentage on those same 112 assertions is only 62%. This decreases to 47% when the frequencies are based on half the assertions and completion is tested on the other half. In contrast, with a leave-four-out testing paradigm, Hinton achieved 88% on unseen assertions, and Quinlan 96%.

Since few algorithms have been applied to the family relations data, MDL/OC was also run on the large soybean data in spite of the fact that it is not relational. Completion was 69% on the training set and 74% on the test set, in contrast to others who have attained close to 100% [Michalski and Chilausky, 1980].

Primarily this poor completion performance is because the MDL principle limits discovered knowledge to that which can be expressed more compactly than the data, and MDL/OC's "rule language" is so simple that only a small class of knowledge can be captured concisely. Better completion was achieved on Family Relations with an encoding scheme that assumed the receiver already knew PERSON1 and RELATION, and so only rewarded features relevant to predicting PERSON2. When features with up to five values were sought, a solution was found that gave 89% and 66% completion on the two testing methods. However the features were a mess. I believe this indicates that the encoding scheme described here does a good job of rewarding only simple features, which are likely to be most easily interpretable. It certainly indicates that completion performance does not always mirror feature interpretability. Given that the goal is enhancing a knowledge base, rather than doing reasoning directly, this is an appropriate tradeoff. Interpretation is hard enough even when learning is very selective.

Indeed, when run on real data from the Cyc KB, the algorithm found features that were largely uninterpretable. Having rejected completion percentage as a measure of feature intuitiveness, and unable to interpret and judge this subjectively, it is still possible to evaluate them on the basis of the amount of compression achieved. For a naive control encoding, I use the MDL/OC scheme with exactly one feature, which segregates the predicates from the individuals. Referring to Figure 1, the ratio for the Family Relations problem is 1262/1601=79%. For one run on Cyc data involving countries' trading partners, it is 95%, which considering how regular the family relations data is, I think significant. With considerable effort, involving automated search for Cyc predicates correlating with the features, I determined that one of them segregated the countries according to economy size. I believe this indicates that even in natural domains, MDL/OC's restricted class of orthogonal features is useful, and that even good features are hard to interpret.

Related Work

Many hierarchical clustering algorithms have been developed, but as mentioned in the problem representation section only those that construct new features are suited to the Family Relations problem, and none exist that can form features as far removed from the given ones as is required here. Additionally, any hierarchical algorithm will suffer data fragmentation. Near the leaves they will have so few training instances that observed statistical regularities are almost entirely spurious. Pagallo and Haussler [1990] suggest a way to recombine identically structured subtrees after the tree is learned. This can be thought of as orthogonalizing. Because orthogonal features are not context sensitive, they 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.

Back-propagation approaches to feature discovery [Hinton, 1986, Miikkulainen and Dyer, 1987] have suffered from the asymmetric treatment of input and output units, and the use of indirect methods like bottlenecks for encouraging good representations, rather than incorporating an explicit declarative characterization of conciseness. This is not an inherent limitation; a back-propagation version of MDL/OC is possible. Declarative evaluation functions are 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. And these techniques are based on the questionable assumption that completion performance is a good measure of feature quality.

Information-theoretic algorithms [Lucassen, 1983, Becker and Hinton, 1989, Galland and Hinton, 1990] avoid the asymmetry of back propagation, but no other algorithm has directly addressed the goals of reducing redundancy or generating a simple completion function. FOIL [Quinlan, 1990] is really not comparable. Quinlan used it to learn intensional definitions of the given predicates in the Family Relations problem. This is much better than the extensional definitions learned here, but he did not construct any new predicates. CIGOL [Muggleton and Buntine, 1988] learns new concepts from relational data, but does not search a large enough space to construct the kinds of features that MDL/OC uses to constrain the tuple elements. On the other hand, it is unnaturally confining for MDL/OC to rely solely on constraints, such as 'the sex of a female-relation must be female.' Much more informative, when available, is data about other relationships in which the people participate. So an algorithm combining MDL/OC's ability to learn radically new concepts with FOIL or CIGOL's

ability to learn rules involving binary predicates would be much more powerful than either approach alone.

Conclusion

This paper has described a new approach to unsupervised discovery or orthogonal features and described its strengths and weaknesses. It is based on a well-motivated declarative description of what good concepts are. The only subjectivity that entered in the derivation is the decision to use an MDL approach at all, and the encoding scheme. The resulting evaluation function has no parameters to adjust. Actually finding solutions that optimize the evaluation function, 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.

If orthogonal features exist for a domain, they are better than hierarchical ones, because they allow maximally general inferences. Algorithms to find them should also be more robust, since data fragmentation is avoided. It is disappointing that completion percentage is not competitive with other algorithms, however this may be a necessary trade-off when seeking simple features. No other algorithm has discovered such clean intuitive features for problems like Family Relations that are not naturally represented as feature-vectors. Therefore I believe the MDL approach, which hopefully can be extended to more expressive and sophisticated encoding schemes, is a promising way to deal with the interpretation problem for learning new representation primitives in a knowledge-based setting.

Still, the difficulty of interpreting feature vectors learned from real data is surprising and frustrating. Although this problem has been recognized and mostly just accepted as inevitable for connectionist learning systems, there was certainly reason to hope that an algorithm designed to find interpretable features, would. Blame on MDL/OC is misplaced, I believe. Rather, new kinds of interpretation tools will be required, either acting in concert with a learning algorithm, or post hoc. Searching for correlations between the discovered concepts and existing ones has been of some help in interpreting features learned from real Cyc data. This problem will doubtless come up for other systems as they evolve beyond local modifications to existing representations.

Acknowledgments 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 help during this last stage reported here. I am also grateful for tutoring in machine learning and suggestions about MDL/OC and the presentation of this paper to Andy Baker, Sue Becker, Guha, Eric Hartman, Jim Keeler, Doug Lenat, Ken Murray, Jeff Rickel, and an anonymous reviewer.

References

[1] 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.

[2] Mark Derthick. The minimum description length principle applied to feature learning and analogical mapping. Technical Report ACT-AI-234-90, MCC, June 1990.

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

[4] 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.

[5] 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.

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

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

[8] 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.

[9] Risto Miikkulainen and Michael Dyer. Building distributed representations without microfeatures. Technical Report UCLA-AI-87-17, UCLA, 1987.

[10] Stephen Muggleton and Wray Buntine. Machine invention of first-order predicates by inverting resolution. In *EWSL-88*, pages 339–352, Tokyo, 1988.

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

[12] J. Ross Quinlan and Ronald L. Rivest. Inferring decision trees using the Minimum Description Length principle. *Information and Computation*, 80:227–248, 1989.

[13] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3), 1990.

[14] Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific Publishing Company, Singapore, 1989.