Multiple-Model Induction in Eleusis

T. G. Dietterich

MLI 80-9

Unrevised version

Papers of the Workshop

on

CURRENT DEVELOPMENTS IN MACHINE LEARNING

Carnegie-Mellon University
Pittsburgh, Pennsylvania

July 16-18, 1980

The support of the Workshop by the Office of Naval Research under grant No. N00014-80-C-0093 is gratefully acknowledged.

Multiple-Model Induction in Eleusis

Thomas G. Dietterich

Department of Computer Science
Stanford University
Stanford, CA 94305

Abstract

The card game Eleusis [1,5] poses an induction problem in which no single model-based generalization space captures all of the desired generalizations of the input data. A program has been developed which solves this problem by searching several different generalization spaces for plausible Eleusis rules. Multiple generalization spaces require multiple data transformation methods for transforming the data into a form suitable for generalization. Examples of the program's operation are given and suggestions are made for further research.

Introduction

Simon and Lea [12] have described induction as a search process which uses two spaces: a space of instances and a space of rules. This characterization of induction is very apt and has recently been extended by Mitchell to take advantage of the general-to-specific ordering of sentences in the generalization space [10].

One of the major goals for future research in induction and problem-solving is the ability to solve problems whose formulation is imprecise—whose generalization spaces have not been provided in advance to the program. The work on the card game Eleusis described herein is a very small step in this direction. Eleusis provides an induction problem for which no single generalization space is sufficient. Instead, several different model-based generalization spaces were identified and implemented. The Eleusis program searches this "meta-space" of models and within each model, conducts a more traditional search for plausible generalizations of the input data. This paper describes the card game Eleusis and the methods used to perform induction in multiple generalization spaces.

A Program for Eleusis

Eleusis (developed by Robert Abbott [1, 5]) is a card game in which players attempt to induce a secret rule invented by the dealer. The secret rule describes a linear sequence of cards. In their turns, the players attempt to extend this sequence by playing additional cards from their hands. The dealer gives no information aside from indicating whether or not each play is correct. Players are penalized for incorrect plays by having additional cards added to their hands. The game ends when a player empties his hand.

A record of the play is maintained as a *layout* (Figure 1) in which the top row, or *mainline*, contains all of the correctly-played cards in sequence. Incorrect cards are placed in *side lines* below the main line card which they follow.

mainline:	3H	9S	4C	JD	2C	10D	8H	7H	2C	5H
sidelines:		JD		AH	AS			10H		
		5D		8H	10S					
				QD						

Rule 1: "If the last card is odd, play black, if the last card is even, play red."

Figure 1. Sample Eleusis Layout (after [1]).

The scoring in Eleusis encourages the dealer to choose rules of intermediate difficulty. The dealer's score is determined by the difference between the highest and lowest scores of the players. Thus, a good rule is one which is easy for some players and hard for others.

This research sought to develop a program which could serve as an intelligent assistant to a human Eleusis player. The program needed to be able to:

- discover rules which plausibly describe the layout,
- accept rules typed by the user and test them against the layout,
- extend the layout by suggesting cards to be played from the player's hand.

The Learning Problem

This paper focusses on the first goal of the Eleusis assistant: the induction of rules which plausibly describe the layout. In the terminology of Simon and Lea, the purpose of the program is to find rules in some generalization space which plausibly describe the layout (which is an element of the instance space). The remainder of this paper discusses what is meant by the terms *plausibly describe* and *some generalization space*, then describes the search process used to search generalization space, and finally presents an extended example.

Descriptions

A rule describes the layout if

- the rule could be used to nondeterministically generate the layout, and
- none of the incorrect plays could be generated using the rule.

In general, a rule describes a whole set of possible layouts just as a grammar describes a set of sentences (a language). For example, Rule 1 describes *any* layout in which all black cards are preceded by odd cards and all red cards are preceded by even cards (and in which all black cards played incorrectly were preceded by even cards and all red cards played incorrectly were preceded by odd cards).

We stress that the layout generation is nondeterministic because of the work done on sequence extrapolation [11] where deterministic descriptions are sought. In Eleusis there are usually several ways to continue the main line.

Plausibility

The conditions for plausibility can be expressed by the following heuristics:

1. **Prefer rules with intermediate degree of complexity.** In Eleusis, Occam's Razor does not always apply. The dealer is unlikely to choose a rule which is extremely simple because it would be too easy to discover. Very complex rules will not be discovered by anyone so the dealer is not likely to choose them either.
2. **Prefer rules with intermediate degree of generality.** The degree of generality—the number of possible layouts which satisfy a rule—is the degree of nondeterminism in the rule. Eleusis rules are typically nondeterministic. Furthermore, rules with low degree of generality lead to many incorrect plays thus rendering them easy to discover. Rules of high generality lead to few incorrect plays and are therefore difficult to discover.
3. **Look for symmetric disjunctions.** Rule 1 is an excellent example of a symmetric disjunctive rule. Most often in Eleusis, the terms of a disjunction define mutually exclusive cases which have some symmetric relationship to each other. The dealer is very likely to choose such rules because they are not too hard—nor too easy—to discover.

These plausibility heuristics are easy to formulate in Eleusis because we have insight into the dealer's goals and behavior. In real world problems, we are rarely so fortunate, but plausibility heuristics can usually still be formulated. Heuristic 3 (symmetry) is built into the model-fitting algorithms described below. The other heuristics are used to evaluate and discard implausible rules once they have been discovered.

Generalization Space—Representation Issues

After examining several Eleusis games, we concluded that a single generalization space would not suffice to describe all Eleusis rules. There are at least three basic classes of descriptions that are typically used in Eleusis: decomposition descriptions, periodic descriptions, and disjunctive normal form descriptions. These can be further combined with segmentation to yield other classes (see [3,4]). Each class of rules can be described in terms of a *model* which describes a logical skelton that can be fleshed out to form a particular rule. (We use the term *model* by analogy with linear regression where the model is a regression polynomial whose coefficients must be determined by fitting it to the data). The three basic models and their parameters are:

- Decomposition. This model specifies that the rule must take the form of an exclusive disjunction of *if-then* rules. The *condition parts* of the rules must refer only to cards prior to the card to be predicted. The *action parts* of the *if-then* rules describe correct plays given that the condition parts are true. The condition parts must be mutually exclusive conjunctive descriptions. The action parts are also conjunctions. Rule 1 fits the decomposition model:

$$\forall_i \text{ odd}(card_{i-1}) \Rightarrow \text{black}(card_i) \vee \text{even}(card_{i-1}) \Rightarrow \text{red}(card_i)$$

  The decomposition model has a *lookback parameter, L*. It indicates that the condition parts of the rules can refer to cards *i-1, i-2, ..., i-L*. *L* must be at least one.

- Periodic. A rule of this model describes the layout as a periodic function. For example, Rule 2 (Figure 2) is a periodic rule. The layout is split into phases according to the length of the proposed period. The periodic model requires that each phase have a conjunctive description.

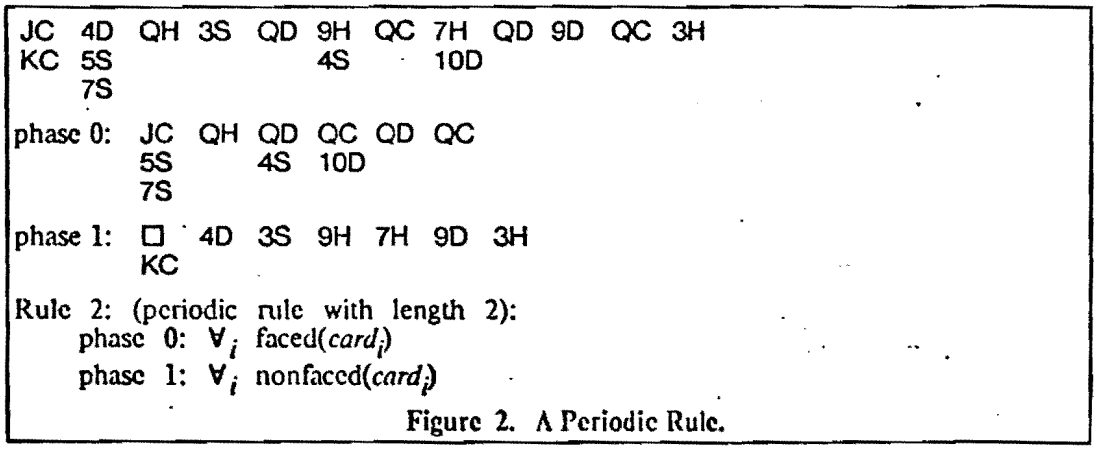

Figure 2. A Periodic Rule.

The periodic model has two parameters: the number of phases, *P*, and the degree of lookback, *L*. *P* tells how many phases the layout should be split into. *L* indicates how far into the past the phase descriptions should go. For instance, the rule

> Play alternating even and odd cards where each even card must be greater than or equal to the last even card and each odd card must be less than or equal to the last odd card

can be written as:

$$\text{phase 0:} \quad \forall i \ \text{even}(\text{card}_i) \wedge \text{value}(\text{card}_i) \geq \text{value}(\text{card}_{i-2})$$
$$\text{phase 1:} \quad \forall i \ \text{odd}(\text{card}_i) \wedge \text{value}(\text{card}_i) \leq \text{value}(\text{card}_{i-2})$$

This rule has a value for *L* of one. Note that we treat each phase as a separate layout with its own indexing (so that $card_{i-1}$ is the card preceding $card_i$ *within this phase*, i.e. actually two cards previous when *P*=2). *L* can be zero as in Rule 2 above.

- Disjunctive Normal Form (DNF) with fewest terms. The A^q algorithm (Michalski [8]) is used to discover rules which have the fewest number of separate conjunctions. The A^q algorithm was given heuristics to guide it towards symmetric, disjoint disjunctive terms. The DNF model also has a lookback parameter L.

Induction Methods

Now that we have identified the generalization spaces and the plausibility criteria, we must show how the search of generalization space is conducted:

1. Choose a model and a parameterization for that model
2. Transform the raw data—the layout—into a set of very specific sentences which have the same logical form as the model. We obtain a set of positive sentences from the main line and a set of negative sentences from the sidelines.
3. Generalize over these sentences to find a plausible sentence which fits the model and covers the positive sentences but not the negative ones.

The approach of representing the instances as very specific points in generalization space is used by many researchers including Michalski [9], Larson [7], Hayes-Roth [6], Vere [14], and Buchanan, *et al.* [2]. Most of the work done on general inductive methods assumes that the instances have been properly encoded in the same representation as the generalizations to be discovered. Problem-specific methods [2, 13], of which Meta-DENDRAL is the most outstanding example, have had to develop sophisticated preprocessing programs to translate the input data (molecule-spectrum pairs in the Meta-DENDRAL case) into a form suitable for generalization (process-evidence pairs in Meta-DENDRAL). In principle, this transformation is unnecessary. We could generate rules in generalization space and then test them against the raw data, but this testing process would be extremely cumbersome. It would usually require repeating this kind of representation transformation for each comparison.

The same argument—that the input data should be transformed before generalization—applies in Eleusis, but instead of one generalization space, we have several—one for each parameterization of each model. Figure 3 shows the shape of this space of models. The Eleusis program performs a nearly exhaustive search of this space of models. One can view this as a process of generate-and-test. We generate each possible model and test it to see how well it fits the data. A further improvement in this program would involve strengthening this weak search method to make it more selective, for example, by testing the simplest models first and searching only until a few satisfactory rules had been found.

An Example

We will show how the program discovers Rule 1 from the layout shown in Figure 1. We will follow one particular branch of the tree shown in Figure 3—that which leads to a decomposition model with $L = 1$. Once this model has been chosen, the layout is converted to the collection of very specific rules examples of which are shown in Figure 4. Notice that the descriptors *color* and *parity* have been added to the data (in addition to the basic descriptors of *suit* and *value*).

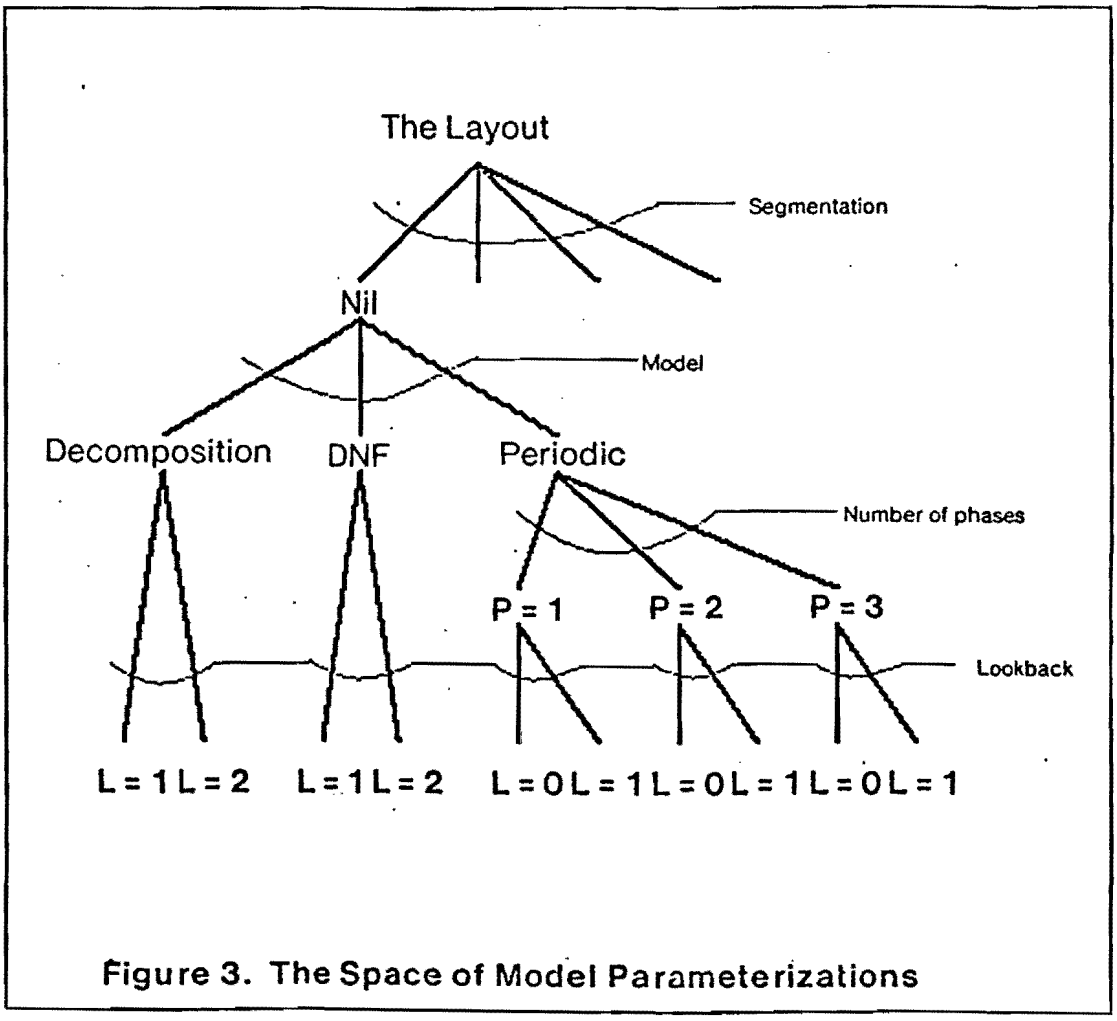

Figure 3. The Space of Model Parameterizations

Positive sentences:

$\text{suit}(\text{card}_{i-1}) = \text{H} \wedge \text{value}(\text{card}_{i-1}) = 3 \wedge \text{parity}(\text{card}_{i-1}) = \text{odd} \wedge \text{color}(\text{card}_{i-1}) = \text{red} \Rightarrow$
$\text{suit}(\text{card}_i) = \text{S} \wedge \text{value}(\text{card}_i) = 9 \wedge \text{parity}(\text{card}_i) = \text{odd} \wedge \text{color}(\text{card}_i) = \text{black}$

$\text{suit}(\text{card}_{i-1}) = \text{S} \wedge \text{value}(\text{card}_{i-1}) = 9 \wedge \text{parity}(\text{card}_{i-1}) = \text{odd} \wedge \text{color}(\text{card}_{i-1}) = \text{black} \Rightarrow$
$\text{suit}(\text{card}_i) = \text{C} \wedge \text{value}(\text{card}_i) = 4 \wedge \text{parity}(\text{card}_i) = \text{even} \wedge \text{color}(\text{card}_i) = \text{black}$

$\text{suit}(\text{card}_{i-1}) = \text{C} \wedge \text{value}(\text{card}_{i-1}) = 4 \wedge \text{parity}(\text{card}_{i-1}) = \text{even} \wedge \text{color}(\text{card}_{i-1}) = \text{black} \Rightarrow$
$\text{suit}(\text{card}_i) = \text{D} \wedge \text{value}(\text{card}_i) = \text{J} \wedge \text{parity}(\text{card}_i) = \text{odd} \wedge \text{color}(\text{card}_i) = \text{red}$

Negative sentences:

$\text{suit}(\text{card}_{i-1}) = \text{S} \wedge \text{value}(\text{card}_{i-1}) = 9 \wedge \text{parity}(\text{card}_{i-1}) = \text{odd} \wedge \text{color}(\text{card}_{i-1}) = \text{black} \Rightarrow$
$\text{suit}(\text{card}_i) = \text{D} \wedge \text{value}(\text{card}_i) = \text{J} \wedge \text{parity}(\text{card}_i) = \text{odd} \wedge \text{color}(\text{card}_i) = \text{red}$

$\text{suit}(\text{card}_{i-1}) = \text{S} \wedge \text{value}(\text{card}_{i-1}) = 9 \wedge \text{parity}(\text{card}_{i-1}) = \text{odd} \wedge \text{color}(\text{card}_{i-1}) = \text{black} \Rightarrow$
$\text{suit}(\text{card}_i) = \text{D} \wedge \text{value}(\text{card}_i) = 5 \wedge \text{parity}(\text{card}_i) = \text{odd} \wedge \text{color}(\text{card}_i) = \text{red}$

$\text{suit}(\text{card}_{i-1}) = \text{D} \wedge \text{value}(\text{card}_{i-1}) = \text{J} \wedge \text{parity}(\text{card}_{i-1}) = \text{odd} \wedge \text{color}(\text{card}_{i-1}) = \text{red} \Rightarrow$
$\text{suit}(\text{card}_i) = \text{H} \wedge \text{value}(\text{card}_i) = \text{A} \wedge \text{parity}(\text{card}_i) = \text{odd} \wedge \text{color}(\text{card}_i) = \text{red}$

Figure 4. Layout Represented as Very Specific Rules

Specialized algorithms have been developed to fit each model to the properly prepared data (see [3,4] for more details). The decomposition algorithm (with $L = 1$) builds "trial decompositions" by attempting to build a rule using each of the attributes of card_{i-1} as the *decomposition variable*. It chooses the best trial decomposition and (if it is consistent with the data) returns it as the induced rule. In particular, when the algorithm builds the trial decomposition based on $\text{parity}(\text{card}_{i-1})$, it obtains the following rule:

$$\begin{aligned}
\forall i \ \text{parity}(\text{card}_{i-1}) = \text{odd} \ \Rightarrow\ & (\text{suit}(\text{card}_i) \in \{\text{C, S}\}) \wedge \\
& (\text{value}(\text{card}_i) \in \{2, 4, 9\}) \wedge \\
& (\text{color}(\text{card}_i) = \text{black}) \\
\vee \ \text{parity}(\text{card}_{i-1}) = \text{even} \ \Rightarrow\ & (\text{suit}(\text{card}_i) \in \{\text{D, H}\}) \wedge \\
& (\text{value}(\text{card}_i) \in \{5, 7, 8, 10, \text{J}\}) \wedge \\
& (\text{color}(\text{card}_i) = \text{red})
\end{aligned}$$

Since this rule is consistent with the data, the program applies rules of generalization to drop the suit and value descriptors. The final, consistent rule is:

$$\begin{aligned}
\forall i \ \text{parity}(\text{card}_{i-1}) = \text{odd} \ &\Rightarrow (\text{color}(\text{card}_i) = \text{black}) \\
\vee \ \text{parity}(\text{card}_{i-1}) = \text{even} \ &\Rightarrow (\text{color}(\text{card}_i) = \text{red})
\end{aligned}$$

Similar techniques are used to discover periodic rules. Michalski's A^q algorithm is used to discover DNF rules.

Conclusion

A learning program can use several models to guide the induction process. For each model, the raw input data must be transformed into a form suitable for model-fitting. Perhaps it is useful to view multiple-model learning as a *recognition* process in which each model "tries" to recognize the input data as being an instance of itself. This perspective may aid the design of systems which can handle large numbers of models. The problem of transforming the data to fit these models will need to be solved first, however, before large numbers of models can be used.

Acknowledgments

Thanks go to R. S. Michalski, my M.S. thesis advisor, for suggesting Eleusis as a domain and for providing numerous ideas including the basic idea for the decomposition algorithm. This research was supported by NSF grant no. MCS-76-22940.

References

[1] Abbott, Robert, "The New Eleusis," Available from Abbott at Box 1175, General Post Office, New York, NY 10001 ($1.00).

[2] Buchanan, B.G., D. H. Smith, W. C. White, R. J. Gritter, E. A. Feigenbaum, J. Lederberg, C. Djerassi, *Journal of the American Chemical Society*, 98 (1976) p. 6168.

[3] Dietterich, Thomas G., "The Methodology of Knowledge Layers for Inducing Descriptions of Sequentially Ordered Events," MS Thesis, Dept. of Com. Sci., Univ. of Illinois, Urbana, October, 1979.

[4] Dietterich, Thomas, G., "Applying General Induction Methods to the Card Game Eleusis," *Proceedings of the First National Conference on Artificial Intelligence*, Stanford University, August, 1980.

[5] Gardner, Martin, "On Playing the New Eleusis, the game that simulates the search for truth," *Scientific American*, 237, October, 1977, pp 18-25.

[6] Hayes-Roth, F., J. McDermott, "An Interference Matching Technique for Inducing Abstractions", *Communications of the ACM*, 21:5, 1978, pp. 401-410.

[7] Larson, J., "Inductive Inference in the Variable Valued Predicate Logic System VL21 : Methodology and Computer Implementation", Rept. No. 869, Dept. of Comp. Sci., Univ. of Ill., Urbana, May 1977.

[8] Michalski, R. S., "Algorithm A^q for the Quasi-Minimal Solution of the Covering Problem," *Archiwum Automatyki i Telemechaniki*, No. 4, Polish Academy of Sciences, 1969 (in Polish).

[9] Michalski, R.S. "Pattern Recognition as Knowledge-Guided Induction," Rept. 927, Dept. of Comp. Sci., Univ. of Ill. Urbana, 1978.

[10] Mitchell, T. M., "Version Spaces: an Approach to Concept Learning," Comp. Sci. Dept. Rept. STAN-CS-78-711, Stanford University, December 1978.

[11] Simon, H. A., and Kotovsky, K., "Human Acquisition of COncepts for Sequential Patterns," *Psychological Review*, 1963, 70, 534-546.

[12] Simon, H. A., and Lea, G., "Problem Solving and Rule Induction: A Unified View," in *Knowledge and Cognition*, L. W. Gregg, ed., Lawrence Erlbaum, Potomac, MD., 1974.

[13] Soloway, E., "Learning = Interpretation + Generalization: a case study in knowledge-directed learning," PhD Thesis, COINS TR 78-13, University of Massachusetts, Amherst, MA., 1978.

[14] Vere, S. A., "Induction of Relational Productions in the Presence of Background Information," In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, MIT, Cambridge, MA., 1977.