Applying General Induction Methods to the Card Game Eleusis

Thomas G. Dietterich

Department of Computer Science
Stanford University
Stanford, CA 94305

Abstract

Research was undertaken with the goal of applying general universally-applicable induction methods to complex real-world problems. The goal was only partially met. The chosen domain—the card game Eleusis—was still somewhat artificial, and the universally-applicable induction methods were found to be lacking in important ways. However, the resulting Eleusis program does show that by using knowledge-based data interpretation and rule evaluation techniques and model-fitting induction techniques, general induction methods can be used to solve complex problems.

Introduction

Work in the area of computer induction is characterized by a continuum from general, universally-applicable methods [5, 6, 7, 9, 10, 12] to specific, problem-oriented methods [2, 8, 11]. The general-purpose methods have been criticized for lacking the power to operate in real-world domains. Problem-oriented methods have been criticized for being too specialized to be applied to any problems outside their original domains. This paper describes an attempt to bridge this gap by applying general-purpose induction algorithms to the problem of inducing secret rules in the card game Eleusis. Further details are available in [3].

A Program for Eleusis

Eleusis (developed by Robert Abbott [1, 4]) 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]).

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.

Although Eleusis is artificial and noise-free, it is sufficiently complex to provide a reasonable test bed for inductive techniques. The development of an intelligent assistant required not only basic induction methods but also extensive deduction techniques for testing rules and extending the layout.

Problems with Existing Induction Methods

While designing the rule-discovery portion of the Eleusis program, existing induction algorithms [5, 6, 7, 9, 10, 12] were examined and found to be lacking in three fundamental ways. The first major problem with some of these algorithms is their emphasis on conjunctive generalizations. Many Eleusis rules are disjunctive. For example, Rule 1 can be written as:

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

The second major problem with these algorithms is that they make implicit assumptions concerning plausible generalizations—assumptions which are not easily modified. Of the algorithms examined, only Mitchell's version space algorithm [10] maintains information concerning all rules consistent with the data (and his algorithm is still oriented toward conjunctive generalization). The algorithms of Hayes-Roth and Vere both seek the most specific rule consistent with the data, while Michalski's A^q algorithm seeks a disjunctive description with the fewest conjunctive terms.

In contrast, the plausibility heuristics for Eleusis are:

1. **Choose rules with intermediate degree of generality.** (Justification: the dealer is unlikely to choose a rule which is overly general because it would be too difficult to discover. Conversely, overly specific rules are easily discovered because they lead to the creation of numerous counter-examples during play.)
2. **Choose disjunctive rules based on symmetry.** (Justification: 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 based on the assumption that the dealer is rational and that he is attempting to maximize his own score (according to the rules of the game). This is an artificial assumption. It is very rare in science that we have such insight into nature. However, in all domains plausibility criteria *must* be available—otherwise, we don't know what we are searching for.)

The third major problem with using general-purpose induction techniques in Eleusis is that the raw data of the Eleusis layout are not in a form suitable for generalization. (Many researchers [2, 11] have pointed out this problem in other domains.) One aspect of this problem is evident in Rule 1: neither *color* nor *parity* is explicit in the representation of the cards. Another difficulty is that the sequential ordering of the cards is implicit in their position in the layout. It must be made explicit in order to discover rules like Rule 1.

Two techniques were developed to address these problems. First, in order to avoid an exhaustive search of rule space and at the same

time avoid the "tunnel vision" of existing algorithms, rule *models* were developed to guide the induction process. Secondly, in order to transform the input data into a form appropriate for generalization, a series of knowledge-based processing layers were created.

Induction by Model-Fitting

By analogy with traditional statistical time-series analysis, the program uses a model-fitting approach to induction. The term *model* denotes a syntactic or functional skeleton which is fleshed out by the induction algorithms to form a rule. In traditional regression analysis, for example, the model is the regression polynomial whose coefficients must be determined by induction from the data. Properly chosen models can strongly constrain the search required for induction. After looking at several Eleusis games, the following models were designed for the Eleusis program:

- **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:

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

- **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.

JC	4D	QH	3S	QD	9H	QC	7H	QD	9D	QC	3H	
KC	5S				4S		10D					
	7S											
phase 0:	JC	QH	QD	QC	QD	QC						
	5S		4S	10D								
	7S											
phase 1:	□	4D	3S	9H	7H	9D	3H					
	KC											

Rule 2: (periodic rule with length 2):
phase 0: $\forall_i$ faced($card_i$)
phase 1: $\forall_i$ nonfaced($card_i$)

Figure 2. A Periodic Rule.

- **Disjunctive Normal Form (DNF)** with fewest terms. The A^q algorithm (Michalski [9]) 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.

By definition, not all Eleusis rules can be represented using these three models. But, these models, when combined with segmentation (see below), cover all but one or two of the Eleusis rules which I have seen.

For each of these models, an algorithm was developed to fit the data to the model. In order to fit the data to the decomposition model, the program must determine which variables to *decompose on*, i.e. which variables to test in the condition part of the rule (Rule 1 decomposes on parity $\in$ {odd, even}). The program must also decide how far into the past this decomposition should apply (i.e. do we look at just the most recent card, or the two most recent cards, ..., etc.). Once the *decomposition variables* and the *degree of lookback* are determined, the algorithm must find a conjunctive description for the action parts of the rules.

The program uses a generate-and-test approach. First, it considers rules which look back only one card, then two cards, and so on until a rule consistent with the data is found. To determine the decomposition variable(s), it generates trial rules by decomposing on each variable in turn and chooses the variable which gives the simplest rule. If the resulting rule is not consistent with the data, the layout is decomposed into sub-layouts based on the chosen variable, and a second decomposition variable is again determined by generating trial decompositions and selecting the simplest. This process is repeated until a rule consistent with the data is found. (This is a beam search with a beam width of 1).

In order to fit the periodic model, the program chooses a length for the period, splits the layout into separate phases, and finds a conjunctive description of each phase. Since the rule is more plausible if the descriptions of each phase are mutually exclusive, the algorithm attempts to remove overlapping conditions in the descriptions of the different phases. Again, a generate-and-test approach is used to generate periodic rules with different length periods (from length 1 upwards) until an acceptable rule is discovered.

The A^q algorithm is used to fit data to the DNF model.

Knowledge-layer Structure

Like many other AI systems, the Eleusis program is structured as a set of layers, or more accurately, rings, based on the kinds of knowledge used to solve the problem (Figure 3). Each layer takes input data from the outside, transforms the data using knowledge appropriate to this layer, performs generalization by searching the space of rules at this level of abstraction, and evaluates the obtained rules. Rules which are sufficiently plausible are returned to the outer layers. Each layer calls the inner layers to perform tasks requiring knowledge appropriate to those inner layers. Figure 4 shows the five layers of the Eleusis program. Notice that in Eleusis, the outermost ring is very specific to Eleusis, while the inner-most rings contain only the general model-fitting induction algorithms. This design is intended to allow the program to be easily applied to similar problems. Since all Eleusis-specific knowledge is in the outer-most two layers, these could be stripped off and replaced by layers which apply different kinds of data transformations to solve different problems (e.g. letter series completion, sequence extrapolation).

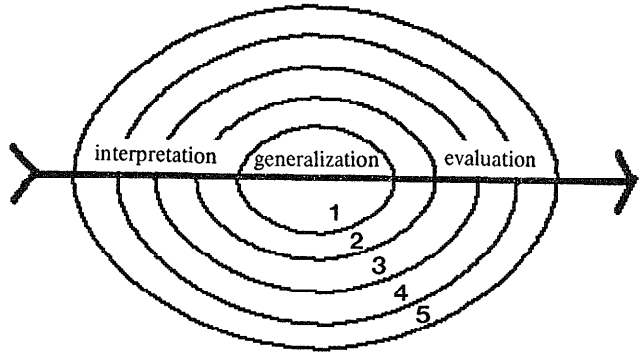

Figure 3. The Knowledge-layer Scheme.

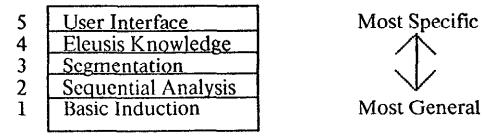

Figure 4. Layered Structure of Eleusis Program.

The five layers function as follows. The outer-most layer provides an interface for the user. Layer 4 transforms the layout by making explicit such things as the *color, parity, prime-ness,* and *faced-ness* of the cards. Layer 3 segments the layout. Segmentation is used to discover rules such as Rule 3 (Figure 5) which involve first splitting up the layout into segments according to some criterion (like constant *color*) and deriving a new layout based on the *lengths* of

these segments. Layer 2 makes the order of the events explicit either by creating "first difference" variables (e.g. Δvalue($card_i$) = value($card_i$) – value($card_{i-1}$)) or by breaking the layout into separate phases (for periodic rules). The result of the preprocessing of layers 5 through 2 is that layer 1 is called with a specific model for which the *degree of lookback* and (optionally) *length of period* have been specified and with a set of unordered events to which the model is to be fitted. Layer 1 actually performs the model-fitting using one of the three model-fitting induction algorithms.

Once the rules have been induced, they are passed back up through the layers for evaluation. Layer 2 evaluates the rules using knowledge about ordering (e.g. guaranteeing that the rule doesn't lead to a dead end). Layer 3 checks that the rules are consistent with the segmentation it performed (in particular, the boundary values cause some problems). Layer 4 evaluates the rules according to the heuristics for plausible rules in Eleusis. Finally, layer 5 prints any rules which survive this evaluation process.

AH	7C	6C KD JH	9S	10H	7H 5S	10D	JC QD	AD	4H 3S	8D	7C

Rule 3: "Play odd-length strings of cards where color is constant within each string."

The *segmented* layout looks like this (color, length):

(red, 1) (black, 3) (red, 3) (black, 1) (red, 3)

Figure 5. A Segmentation-based Rule.

Results

The program works well. The three rule models, when combined with segmentation, span a search space of roughly 10^{183} possible rules (several control parameters affect the size of this space). The program generates and tests roughly 19 different parameterizations of the three models in order to choose three to five plausible rules. It runs quite quickly (less than seven seconds, on a Cyber 175, in the worst case so far). The rules developed are similar to those invented by humans playing the same games (15 complete games have been analyzed).

Conclusion

General induction techniques can be used to solve complex learning tasks, but they form only part of the solution. In the Eleusis domain, data interpretation, rule evaluation, and model-directed induction were all required to develop a satisfactory program.

A degree of generality was obtained by segregating the functions of the program into layers according to the generality of the knowledge they required. This should allow the program to be applied to similar tasks merely by "peeling off" and replacing its outer layers.

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. Thanks also to David Shur for proofreading this paper. 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] Gardner, Martin, "On Playing the New Eleusis, the game that simulates the search for truth," *Scientific American*, 237, October, 1977, pp 18-25.

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

[6] Hunt, E.B., *Experiments in Induction*, Academic Press, 1966.

[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] Lenat, D., "AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search," Comp. Sci. Dept., Rept. STAN-CS-76-570, Stanford University, July 1976.

[9] 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).

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

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

[12] 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.