Two Short Papers on Machine Learning

Nicholas S. Flann and Thomas G. Dietterich
Department of Computer Science
Oregon State University
Corvallis, Oregon 97331

86-30-3

Two Short Papers on Machine Learning

Nicholas S. Flann and Thomas G. Dietterich
Department of Computer Science
Oregon State University
Corvallis, OR 97331

Abstract

This technical report reprints two articles that appeared in *Proceedings of the Third International Machine Learning Workshop* at Skytop, Pennsylvania, June 24-26, 1985. The first paper, *The EG Project: Recent Progress*, summarizes work on the EG project, which is investigating the role of active experimentation in aiding machine learning programs. The second paper, *Exploiting Functional Vocabularies to Learn Structural Descriptions* describes work on the problem of developing computer programs that automatically construct their own representational vocabulary.

The EG Project: Recent Progress

Thomas G. Dietterich
Department of Computer Science
Oregon State University
Corvallis, OR 97331

1 Introduction

The long term goal of the EG project is to construct, implement, and evaluate a model of scientific inquiry. Figure 1 shows the model of scientific inquiry that we have developed to date.

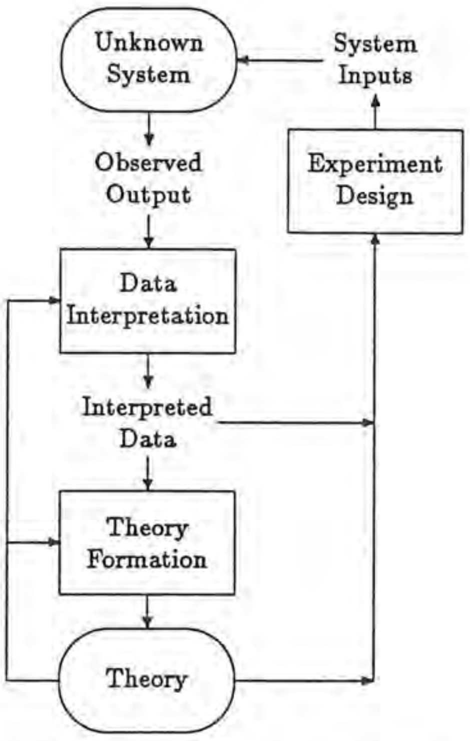

Figure 1: Simple model of scientific inference

According to this model, the goal of a scientist is to construct an accurate theory of an unknown system. There are three main processes that work together to achieve this goal: data interpretation, theory formation, and experiment design. The data interpretation process takes observed behavior of the unknown system (i.e., "raw" data) and applies the current theory (or theories) to develop a set of interpretations of the data. The theory formation process modifies the current theory (or theories) so that they are consistent with this interpreted data. During the data interpretation and theory formation processes, whenever ambiguity is encountered the scientist faces a choice. He or she may either proceed with data interpretation and theory formation—maintaining alternative interpretations and theories as necessary—or the scientist can design an experiment that would resolve the ambiguity. It is the task of the experiment design process to design and carry out such experiments.

In order to investigate and refine this model, we are implementing it in a specific task domain. The specific learning task is the problem of forming theories about the file system commands of the UNIX operating system. The goal is to develop a computer program that starts with knowledge of two of the UNIX commands (the directory listing command and the command to display the contents of a file) and proceeds, through experimentation and observation, to develop theories of how thirteen different commands work. This task has been discussed in detail elsewhere [2,3], so we will simply describe how the components of Figure 1 are instantiated in this domain. The unknown system, of course, is the UNIX operating system. It accepts typed text as inputs and produces strings of text as outputs. It also contains state information (e.g., the file system, current working directory, etc.). A theory in this domain consists of 13 procedures—one for each UNIX command. The task of data interpretation is to apply existing theories about some of the commands to infer as much as possible about the current state of UNIX (see below). This can be viewed as applying the current (partial) theory to "parse" the raw data. From the parsing process, EG obtains an input/output pair (training instance) for each command that was executed. The theory formation process takes these input/output pairs and uses them to guide the modification of the current set of theories. Hence, it constitutes a kind of procedure induction from I/O pairs. The experiment design process can be invoked whenever ambiguities arise during data interpretation and theory formation. It designs and executes experiments to gather additional data.

Since the start of this project (in 1983), we have focused our efforts primarily on the process of data interpretation. The data interpretation task can be formally defined by reference to the deductive-nomological (D-N) model of scientific explanation. According to this model, a body of data D can be *explained* by a theory T and a set of initial conditions C if from T and C you can deductively infer D. Data interpretation is the process of finding a set of initial conditions C given theory T and data D. It can be shown that C is a deductive consequence of T and D, so data interpretation can be be performed by a deductive process.

For example, suppose that the EG program wants to form a theory of the `rm` command. It might have the following interaction with UNIX in which a directory listing is obtained for the `/csm/eg` directory before and after the execution of the unknown `rm` command:

```
1: ls /csm/eg
file1
file2
file3
2: rm /csm/eg/file2
3: ls /csm/eg
file1
file3
4:
```

By applying its theory of the directory listing (`ls`) command, EG can make the following inferences based on the first `ls` invocation: (a) the root directory contains a directory named `csm`; (b) the `csm` directory contains a directory named eg; (c) the eg directory contains three files `file1`, `file2`, and `file3`. Similarly, the second `ls` command can be interpreted to infer that after the `rm`, the eg directory contains only two files: `file1` and `file2`. To obtain each of these interpretations, EG takes its given theory T of `ls` and finds a set of initial conditions C (i.e., facts about the existence of various files and directories) such that the observed data D (the printed output of the `ls`) can be explained as a deductive consequence of T and C.

Once these interpretations have been constructed, EG has a complete I/O pair for the unknown `rm` command. This I/O pair is passed to the theory formation process, which will construct one or more theories to explain how `file2` could have disappeared.

From this example, we can see that data interpretation plays a role in scientific inquiry analogous to the role of goal regression in planning—both techniques support the incremental development of composite structures (either theories or plans). The fundamental idea behind incremental planning is to "break off" a piece of the overall goal and construct a plan for that piece. Once this is done, goal regression techniques [11] can be applied to "push" the original goal "through" the part of the plan that has already been constructed. This regressed version of the goal specifies the subgoals that remain to be achieved.

In an analogous fashion, data interpretation serves to "push" the observed data through the parts of the theory that have already been developed. The interpreted data then serve to guide further theory formation. This residual theory formation problem is easier to solve than the original task of inferring a theory based on the raw data alone.

Dietterich [3] describes the implemented system PRE that applies constraint propagation techniques to the data interpretation problem. PRE (Program Reasoning Engine) employs the "deep plans" representation developed in the Programmer's Apprentice project to represent theories as constraint networks. The observed printed output from UNIX is asserted on these networks, and then constraint propagation methods are applied to propagate the outputs *backwards* through the theory to infer possible input states that would have caused the outputs. Hence, PRE solves the data interpretation problem by "backwards execution" of programs. PRE is implemented in Interlisp-D on the Xerox 1108 lisp machine.

Now that we have implemented the data interpretation component of our model, we are now focusing on the experiment design component. Previous work in this area has included a series of "idea" papers that discuss the value of experimentation (e.g., [1,4,10]) and a collection of computer programs that perform some kind of instance selection or experimentation (e.g., [5,6,7,8,9]). We are extending this work in two directions. First, we are investigating experimentation and theory formation in domains where the "unknown system" under study is a system that contains *state information*. Second, in addition to developing effective experiment design methods, we are attempting to understand the space of possible methods and strategies. In particular, we are exploring the ways in which experimentation strategies introduce additional biases into the theory-formation process.

2 References

[1] Buchanan, B., Mitchell, T., Smith, R., Johnson, C. in J. Belzer, A. Holzman, and A. Kent (eds.), *Encyclopedia of computer science and technology*, New York: Marcel Dekker, 24–51, 1979.

[2] Dietterich, T. G., *AAAI-84*, 96–100, 1984.

[3] Dietterich, T. G., "Constraint propagation techniques for theory-driven data interpretation," Rep. No. STAN-CS-84-1030, Stanford Univ., 1984.

[4] Dietterich, T., Buchanan, B. in Rissland, E., Arbib, M., and Selfridge, O., *Adaptive Control of Ill-defined Systems*, Plenum, 1983.

[5] Kibler, D., Porter, B., *IJCAI-83*, 415–418, 1983.

[6] Lenat, D., *Art. Int.*, 21:31–60, 1983.

[7] Mitchell, T., Utgoff, P., Banerji, R., in Michalski, R., Carbonell, J., and Mitchell, T. (eds.), *Machine Learning*, Palo Alto: Tioga, 1983.

[8] Quinlan, J., in Michalski, R., Carbonell, J., and Mitchell, T. (eds.), *Machine Learning*, Palo Alto: Tioga, 1983.

[9] Shapiro, E., Rep. No. 192, Yale University, 1982.

[10] Simon, H. A., Lea, G., in L. Gregg (ed.), *Knowledge and Cognition*, Hillsdale, N.J.: Lawrence Erlbaum, 105–127, 1974.

[11] Waldinger, R. J., in E. Elcock, D. Michie (eds.) *Machine Intelligence 8*, New York: Halstead/Wiley, 1977.

Exploiting Functional Vocabularies to Learn Structural Descriptions

Nicholas S. Flann and Thomas G. Dietterich
Department of Computer Science
Oregon State University
Corvallis, OR 97331

1 Introduction

In the idea paper entitled "Learning Meaning," Minsky [3] stresses the importance of maintaining different representations of knowledge, each suited to different tasks. For example, a system designed to recognize examples of cups on a table would do well to represent its knowledge as descriptions of observable features and structures. In contrast, a planning system employing cups to achieve goals would require a representation describing the purpose and function of cups. When we turn from the issue of *employing* a description of a cup to the task of *learning* such a description, it is not immediately obvious what vocabulary should be used. One approach might be to choose the vocabulary appropriate for the performance task (i.e., structural descriptions for recognition, functional descriptions for planning, etc.). This approach has been pursued, e.g., by Winston [10], Buchanan & Mitchell [1], Quinlan [7], and Minton [4]. In the case of Winston's ARCH learner and Buchanan & Mitchell's Meta-DENDRAL system, this approach worked well because good structural vocabularies were available. However, Quinlan and Minton confronted much more difficult problems in constructing structural vocabularies that concisely captured the desired game-playing concepts. Quinlan, for example, spent two man months developing the vocabulary for the concept of "lost-in-3-ply."

The difficulty that these researchers encountered is that in addition to considering the representation requirements of the performance task, one must consider the representation requirements of the learning process. By their very nature, inductive learning systems must operate on the syntactic form of the descriptions being learned. The biases of such systems are stated in syntactic—hence, vocabulary-specific—terms. To meet the combined requirements of the performance task and the learning process, the vocabulary must be very carefully engineered!

An alternative to this "single-vocabulary" approach is to employ two different vocabularies: one for learning and one for performance. In many domains, the desired concept can be expressed very succinctly in one vocabulary, and yet this vocabulary cannot be efficiently employed to perform the desired task. For example, Winston et al. [9] have shown that the concept of "cup" can be expressed very succinctly in a functional vocabulary, and yet it is difficult to use such a description to recognize pictures of cups. Figure 1 summarizes this approach to acquiring an efficient structural description by first learning a succinct functional description. In this approach, the given examples (presented in a simple structural vocabulary) are converted into functional examples through a process of "envisionment." Then the functional examples are inductively generalized to obtain a functional concept description. Finally, this concept description is converted to a structural concept description through a compilation process.

We have developed a program (named Wyl) that learns simple concepts in chess and checkers via this "two-vocabulary" approach. We have chosen board games such as chess and checkers because there are many interesting concepts that have simple functional characterizations (e.g., "trap," "skewer," "fork," "lost-in-2-ply") and yet have quite complex structural definitions. Wyl has been applied to learn definitions for "trap" and "move-to-trap" in checkers and "skewer" in chess. We illustrate Wyl's functioning with "trap".

2 An Example: "Trap"

2.1 Envisionment

Wyl starts with a given structural training instance (i.e., board position), which it is told is an instance of "trap" (Figure 2). To convert this into a functional instance, it conducts a forward minimax search according to the rules of checkers looking for "interesting" positions (e.g., win, loss, draw, loss of important piece, etc.). The result of this search process is a game tree (A) in which each move is tagged with its computed consequences, returned from the min/max search. The tree is then traversed to construct a proof of the backed-up value of the root. The "theorem" to be proved always takes the form of a series of nested quantifiers (e.g., for a *win*, the proof states that "there exists a move for me such that for all of my opponent's moves ... I win." For a *loss*, the proof states that "for all moves I make there exist moves for my opponent such that ... I lose."). The nodes of the tree are labeled with the appropriate quantifiers, and branches of the tree that do not contribute to the proof are eliminated. This yields tree (B), which in Mitchell's terms [6] could be called an "explanation tree" for the computed result.

2.2 Generalization

Tree (B) can be viewed as providing two positive functional training instances of a loss for the first player. The generalization step "compresses" these two parallel branches of the tree to obtain tree (C). Two biases are applied to drive this generalization. The first bias is the familiar bias toward maximally specific conjunctive generalizations. The final functional concept definition must be a conjunction (with interspersed nested quantifiers as dictated by tree (B)). To implement this bias, two generalization rules are applied: (i) constants are changed to variables and (ii) the alternative branches of the tree are changed to a universal quantifier. This second generalization rule is equivalent to the inverse-enumeration rule:

$$P(C1) \wedge P(C2) \wedge \ldots \wedge P(Cn) \Rightarrow \forall c P(c)$$

The second bias employed by Wyl states that "There are no coincidences." More concretely, if the same constant appears at two different points within a single training instance, it is asserted that those two different points are *necessarily* equal. Figure 2 demonstrates this bias. In the left branch of tree (A), the white piece first moves to square s6. Then the red piece in square s10 captures the piece in square s6. The "no coincidences" bias says that these two occurances of s6 were necessarily identical. Hence, when they are matched against the right branch of tree (A) to s7, they are both generalized to the *same* variable *square1*. The final functional concept description states that a "trap" is a situation in which your opponent has a single piece (at s10) that is able to capture your piece (at s2) no matter what move you make. This is an overly specific version of trap. A second structural training instance must be presented to Wyl before the correct definition of "trap" is found.

2.3 Compilation

The third stage of the learning process is termed "operationalization" by Mostow and Keller. Several techniques have been developed for performing this process. Keller [2] and Mostow [5] apply program transformation methods to transform the functional description into a structural description. Utgoff [8] and Minton [4] apply constraint back-propagation. An alternative approach of enumeration and compaction was explored in Wyl as no structural language was available. First, an intelligent generator is applied to generate all possible board positions consistent with the generalized functional description. In the "trap" case, 146 possible positions are generated. Second, two algorithms are applied to compress this set of 146 positions into a disjunction of 12 general descriptions. One algorithm identifies common internal disjunctions over squares and move directions to define terms such as "center," "north," and "double-corner side." The second algorithm invents relational terms as compositions of primitive relationships. For example, the term north-2-squares(*square1 square2*) is defined as a conjunction of nw(*square1 square3*) and ne(*square3 square2*).

3 Conclusion

The approach of employing two vocabularies (structural and functional) and performing induction only in the functional vocabulary has significant advantages for acquiring efficient structural descriptions of concepts that have succinct functional descriptions. First, fewer examples are required to learn the concept. Second, the bias built into the learning program is very simple (maximally-specific conjunctive generalization). Third and last, the learning system starts with less builtin knowledge and hence requires less domain-specific engineering.

4 References

[1] Buchanan, B. G. and Mitchell, T. M., *Pattern-Directed Inference Systems*, Waterman, D. A. and Hayes-Roth, F. (Eds.), Academic Press, New York, 1978.

[2] Keller, R. M., *AAAI-83*, 1983.

[3] Minsky, M. "Learning Meaning", Unpublished manuscript, Massachusetts Institute of Technology, (1982).

[4] Minton, S. *AAAI-84*, 1984.

[5] Mostow, D. J., *Machine Learning*, Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (Eds.), Tioga Press, Palo Alto, 1982.

[6] Mitchell, T., *IJCAI-83*, 1139-1151, 1983.

[7] Quinlan, J. R., *Machine Learning* Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (Eds.), Tioga Press, Palo Alto, 1982.

[8] Utgoff, P. E. and Mitchell, T. M., *AAAI-82*, 1982

[9] Winston, P.,Binford, T., Katz, B. and Lowry, M. *AAAI-83*, 1983.

[10] Winston, P. H., *The Psychology of Computer Vision*, Winston, P. H. (Ed.), McGraw Hill, New York, ch. 5, 1975.

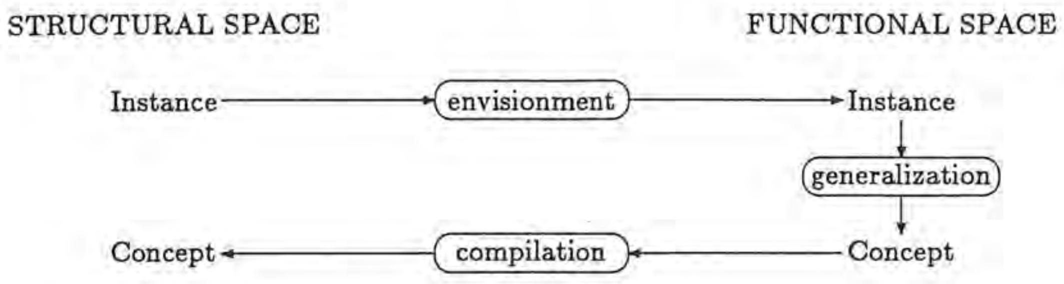

Figure 1: Functional Learning Scheme

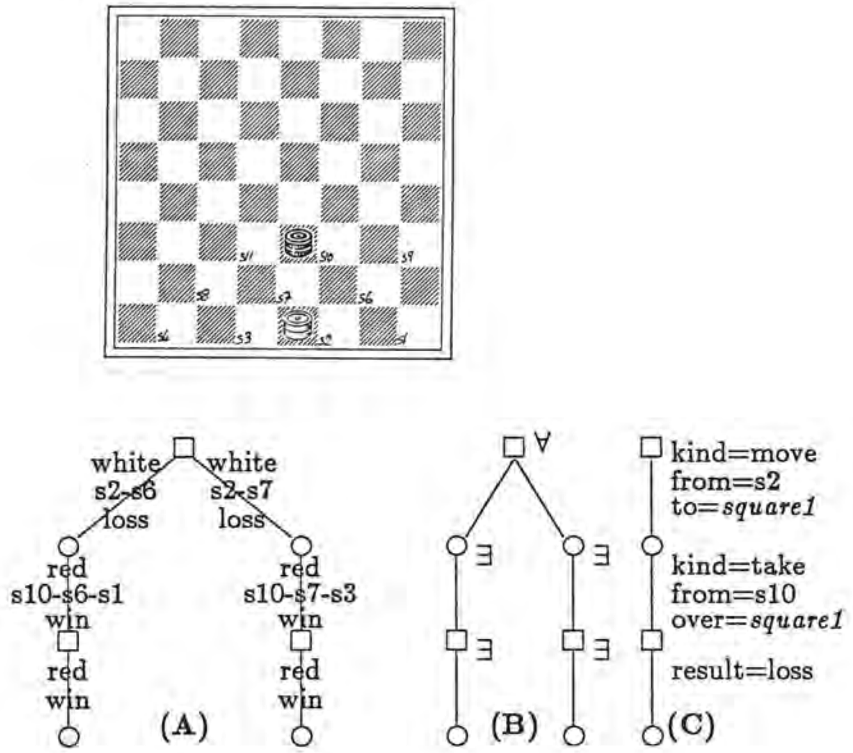

Figure 2: Checkers concept trap