

Techniques and Methodology

Learning from Solution Paths: An Approach to the Credit Assignment Problem

Derek Sleeman

*Department of Computer Studies
University of Leeds
Leeds LS2 9JT
United Kingdom*

Pat Langley

*The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213*

Tom M. Mitchell

*Department of Computer Science
Rutgers University
New Brunswick, New Jersey 08903*

Abstract

In this article we discuss a method for learning useful conditions on the application of operators during heuristic search. Since learning is not attempted until a complete solution path has been found for a problem, credit for correct moves and blame for incorrect moves is easily assigned. We review four learning systems that have incorporated similar techniques to learn in the domains of algebra, symbolic integration, and puzzle-solving. We conclude that the basic approach of learning from solution paths can be applied to any situation in which problems can be solved by sequential search. Finally, we examine some potential difficulties that may arise in more complex domains, and suggest some possible extensions for dealing with them.

PEOPLE LEARN FROM EXPERIENCE, and for the past 25 years, Artificial Intelligence researchers have been attempting to replicate this process. In this article we focus on learning in domains where *search* is involved. Furthermore, we will restrict our attention to cases in which the *legal* operators for a task are known, and the learning task is to determine the conditions under which those operators can be *usefully* applied. Once such a set of heuristically useful conditions has been discovered, search will be directed down profitable

paths, and thus greatly reduced.

In the following section we will examine four systems that learn useful conditions on operators by examining solution paths. We shall see that this process can be divided into three components. Firstly, one must be able to find one or more solution paths to the goal state. Secondly, one must be able to assign credit when a correct operator is used and assign blame when an incorrect operator is selected. Thirdly, one must be able to modify the responsible operators to reinforce positive behavior and to prevent errors in the future. After describing the learning systems, we explore their common features and consider extensions of the basic approach to more complex domains.¹

Recent Research on Strategy Learning

The past few years have seen the emergence of a new approach to learning strategies from sample solutions and by experimentation. Below we summarize four research efforts that have remarkable overlap, despite the fact that they occurred in relative isolation. Although the first workers in this area did not seem to realize the full implication of their techniques, the basic ideas were nevertheless present. After

We would like to thank Jaime Carbonell and Hans Berliner for helpful comments on an earlier version of this article. This article was written during Derek Sleeman's visit to Carnegie-Mellon University, under support of the Sloan Foundation. Tom Mitchell was supported by Grant MCS80-08889 from the National Science Foundation, while Pat Langley was supported by NSF Grant SPI-7918266.

¹Readers interested in the wider aspects of learning are referred to Doyle and London (1980), Mitchell, Carbonell, and Michalski (1981), and Michalski, Carbonell, and Mitchell (in press).

discussing each of the systems individually, we attempt to formulate a general method for learning from solution paths in terms of their commonalities.

Brazdil's ELM Brazdil (1978) has discussed ELM, a PROLOG program that learns from sample solutions in the domain of simple algebra and arithmetic. The system starts with a set of operators for associativity, for adding the same number to both sides, and so forth. It is then given as input a sequence of practice problems, along with their solutions. For each problem, ELM goes through each step of the solution, comparing the step it would have made with the corresponding step of the known solution. Since ELM has no priority ordering for its operators at the outset, it tries all operators applicable to the current problem state. Only one operator application agrees with the solution trace, so the corresponding rule is given priority over its competitors; in the future, this rule is selected in preference to the others. In this way, ELM establishes a partial ordering on its set of operators. Difficulties arise when one problem suggests a certain ordering, and another problem suggests a different one. In such cases, Brazdil's system creates more constrained versions of the competing operators with additional conditions on their application. The new conditions are selected by finding predicates that were true when the operator should have been applied, but false when another operator should have been preferred. The new operators are added to the priority ordering above the rules from which they were generated.

Thus, ELM addresses two of the issues identified earlier, since it assigns credit and blame to particular operators and since it modifies the condition parts of the rules appropriately. However, Brazdil's system does not initially attempt to find its own solution paths.

Neves' ALEX. Neves (1978) has described ALEX, another system that learns algebra procedures from examples. The program is stated as an adaptive production system, and learning is accomplished through the addition of new condition-action rules. ALEX creates two types of productions. The first of these recognizes familiar differences between successive states in a problem. The second type of rule is responsible for proposing useful operators to apply, in order to transform one state into another. Although ALEX's standard learning mode expects solutions in which all the steps are explicitly given, the system has the ability to fall back on means-ends analysis to fill in missing steps in a sample problem. Once the program has successfully solved a problem with this technique, it processes the completed solution as if it had been provided by the user.

Thus, Neves' system is able to discover its own solution paths and learn operators, but avoids the credit assignment problem and the *revision* of faulty operators. In contrast, the next two systems we will consider have adequately dealt with all three issues.

Mitchell's LEX. Mitchell, Utgoff, Nudel, and Banerji (1981) have described LEX, a computer program that acquires problem solving heuristics in the domain of symbolic

integration. This system starts with legal operators for transforming mathematical expressions, and learns heuristics (i.e., conditions under which operators should be applied) in three steps:

1. using heuristics that it has formulated previously in order to solve practice problems.
2. analyzing the steps it performed in solving the problem to isolate positive and negative examples of useful search steps
3. proposing and refining domain-specific heuristics to improve its performance on subsequent problems

The credit assignment method used in this system isolates negative instances as search steps that lead from some state on the solution path to some state off the solution path. However, before labeling such steps as negative instances, LEX selectively expands part of the search to determine whether an alternative solution could be derived along some path involving the "deviant" search step.

In formulating heuristics, LEX maintains a *version space* for each proposed heuristic; that is, it maintains a range of alternative statements of the heuristic that are plausible given the observed data. As new positive and negative examples are encountered, the version space of that heuristic is refined using the Candidate Elimination Algorithm (Mitchell, 1978). The representation of partially formulated heuristics in terms of their version space has been found to be important for solving subsequent practice problems. One advantage of this representation is that the problem solver can determine the *relevance* of a heuristic in a particular situation by determining what proportion of the alternative versions of the heuristic match the situation. Thus, the problem solver can make use of heuristics that are still under revision, while minimizing the likelihood of being misled by an incorrect heuristic.

Langley's SAGE. Langley (1981) has described SAGE, a strategy learning program stated as an adaptive production system and implemented in the PRISM formalism (Langley and Neches, 1981). Like Brazdil's system, SAGE begins with a set of overly general operators for solving a problem. And like Neves' program, SAGE attempts to find its own solution path using depth-first search. Having found a solution path, the system tries the same problem a second time. When an operator is applied incorrectly, SAGE compares the most recent correct application of that operator to the current faulty one in search of differences.

Differences consist of propositions which were present in memory during one application but not during the other. These may relate to the current problem state, to previous moves that have been made, or to a combination of both types of information. Such a difference can be composed of a single proposition or a conjunction of propositions. If a set of propositions are found to have been present during the correct application but not during the errorful one, these facts are included as additional conditions on a new version of the faulty production, much as in Brazdil's system. If the

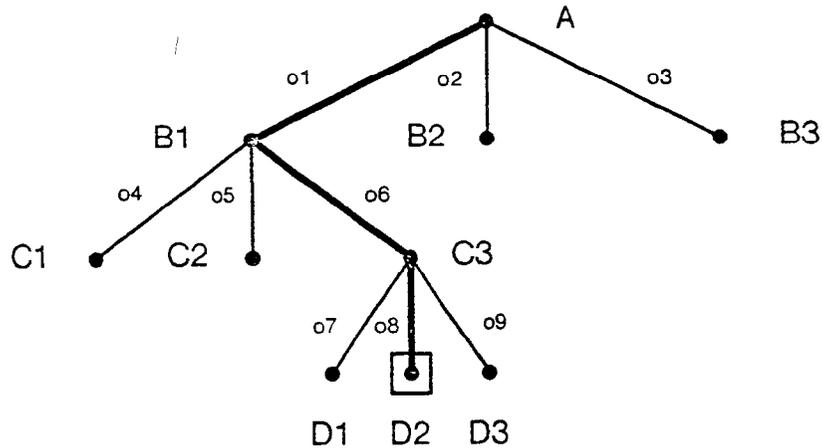


Figure 1. A sample search tree.

relevant elements were present during the incorrect application, they are included as a set of negated conditions on the variant rule (i.e., the new rule would be applicable only if the new facts were not all true). All possible differences are found, and the resulting variant rules are constructed. The initial rule is retained, and it can lead to additional errors in the future.

Whenever SAGE regenerates an existing rule, the strength of that rule is increased. This means that since useful rules tend to be learned more often than spurious ones, they come to be preferred. Eventually, the strength of the correct variants exceed the strength of the original operators. If a variant production is still overly general, it leads to its own errors and other rules are created with still more conditions. However, SAGE eventually arrives at a set of rules that allow it to solve the current problem with no errors. When this occurs, the system is satisfied that it has learned enough and asks for a new problem for which its refined operators may or may not be useful. SAGE has discovered useful operators in the domains of simple algebra, the slide-jump puzzle, and seriation.

An overview of the approach. Taken together, the above systems suggest an approach to strategy learning that is both general and powerful. Starting with a set of legal operators for solving a class of problems, one can solve sample tasks using weak, general methods such as heuristic search and means-ends analysis.² At this point, there are several possible strategies for learning useful conditions. One of these, followed by Langley's system, may be labeled "optimistic" or "uncautious." Another, followed by Mitchell's program, may be described as "cautious." In both cases the search tree is examined to determine *correct* applications of operators which are treated as positive instances, while *incorrect* applications are treated as negative instances that should be avoided in the future.

²This analysis assumes that the cost of a path is given by its length. This analysis can be easily extended to accommodate variable-cost paths.

Given a partial search tree, these two strategies would, in general, provide a different set of training instances. For example, consider the search tree presented in Figure 1. Given that one has discovered that the node D2 is the goal and that the path A B1 C3 D2 is a solution path, then the optimistic algorithm would infer that the particular instantiations of:

op1 at A, op6 at B1, and op8 at C3 are *positive* instances;

and the particular instantiations of:

op2 at A, op3 at A, op4 at B1, op5 at B1, and op7 at C3, and op9 at C3 are *negative* instances.

In contrast, the cautious algorithm would use the same set of positive instances, but it would only decide on the negative set of instances after having further investigated the cost associated with the relevant sub-trees.

Two additional mechanisms are useful. Firstly, as the data handled by the the learning mechanisms is inherently errorful, it is essential (particularly for the less cautious algorithm) that the algorithms employed should be able to cope with noisy data. Both Mitchell's and Langley's systems do have some ability to deal with errorful data. Secondly, the algorithm could attempt to *extend* parts of the solution tree in order to provide additional training instances. For example, for the search tree of Figure 1, LEX would ask the problem solver to try to "solve" node B3. The problem solver is given a bound on the search effort to be spent on this task, which is based upon the effort originally expended to find the currently known solution.

Let us for a moment consider what the situation would be if one had access to the *complete* search tree. Such a situation would have three advantages. Firstly, it would provide the learning algorithm with many more positive and negative instances. (In these cases the "optimistic" and the "cautious" algorithms would infer the same training sets.) Secondly, the data would be less noisy (this particularly affects the "optimistic" strategy). Thirdly, it would then be possible to learn with respect to the *optimum* path(s), as opposed to a *possible* path. (Indeed, it might be sensible to "initialize"

the learning process by choosing a simple, yet characteristic, problem which could be explored exhaustively.)

Although, the particular learning mechanisms that have been employed are interesting in themselves, we will not focus on them here. The central insight that makes such learning possible is that once a solution path has been found, the credit assignment problem becomes tractable, and one can then infer a set of positive and negative instances to drive the learning process. The resulting operators will incorporate additional domain-specific knowledge to direct future search in profitable directions. This notion is deceptively simple, but it provides the basis for a general approach to strategy learning. (Note that many earlier approaches tried to decide what was wrong with a path which led to backtracking *before* a complete solution path had been discovered.)

Generality of the Approach

The approach to learning proposed in the previous section would seem to be applicable to any domain in which problems can be solved by the (sequential) application of operators.³ Fortunately, much of Artificial Intelligence research has been devoted to casting challenging tasks in this framework (Newell and Simon, 1976). Examples include solving simple puzzles such as the Tower of Hanoi, simplifying expressions in algebra and calculus, inducing rules from examples, proving theorems in logic and geometry, and playing two-person games like checkers and chess. We would like to emphasize that the proposed method of learning from solution paths can be applied to any of these domains, and that it provides a general approach to learning that deserves more attention in the future.

However, we would *not* claim that the method is capable of learning all there is to learn in any domain. The proposed technique lets one discover heuristically useful conditions on the application of legal operators, but one can certainly think of other interesting dimensions of the learning process that we have not addressed. For example, Anzai (1978) has examined the learning of subgoals in the Tower of Hanoi task. Neches (1981) has modeled the creation of new and more efficient operators in the arithmetic domain. And Iba (personal communication) has considered the discovery of macro-operators in puzzle-solving tasks. Still, we feel that the determination of conditions has an important role to play in any field where search is involved. Having argued for the generality of the method, let us now turn to its potential limitations.

Large search spaces. The methods described above emphasize the need to determine solution paths before credit assignment can be carried out effectively. However, some

search spaces may be so large that weak methods are insufficient to produce *any* solution paths. Let us consider what can still be accomplished in such situations. In the case where the search space is large, it may be necessary to “focus” the problem solver on simpler problems, or at least ones which have fewer steps in their solutions, and to evolve appropriate heuristics for such problems before considering more complex ones. Alternatively, the investigator could provide the learning mechanism with a solution *trace* for the “unsolvable” problems, as Brazdil did for his learning system. A less extreme form of this approach would be to provide *subgoals* along the solution path, so the complex problems could be broken down into solvable sub-problems.

Poor moves and catastrophic moves. So far, we have distinguished only between solution paths and failure paths. However, in some circumstances it may be useful to distinguish between poor (yet failure) paths and catastrophic failures. For example, in chess it might be useful to distinguish between moves which led to the loss of a pawn and ones that led to the loss of a queen. If such a distinction is desirable, a further set of heuristics might be created by providing an algorithm that treats “poor” nodes as “good” and catastrophic nodes as “bad.” Indeed, if necessary a whole range of such discriminations could be created using the mechanisms outlined here.

Application to AND/OR trees. Another possible limitation of the approach relates to games involving AND/OR trees. As in the case for OR trees, a complete expansion of the search tree is necessary in order to assure completely reliable credit assignment. However, the same approaches used for approximating the search for OR trees can be applied here.

In such games, the actions of the opponents may be unpredictable. In practice then, one would be learning to improve the heuristics for operators with respect to a “black box” system whose strategy would *not* be effected by the “refined” heuristics being learned. If, for some reason, it was decided that the “black-box opponent” should have access to the revised rules, then the solution path might change as a result of an earlier refinement to a rule-heuristic. Thus, one would need to repeat the learning cycle *until* no further changes in the rule conditions *and* the solution path were noted.

Conclusions

In this article we have discussed a method for learning from solution paths that has been used to revise the conditions on operators from a variety of domains, including algebra, symbolic integration, and puzzle-solving. This approach provides an elegant yet powerful solution to the credit assignment problem, and seems applicable to any problem that can be solved with sequential search. Of course, the discovery of conditions is not the only aspect of learning in such situations, but it does appear to be an important component of the overall process. And although difficulties arise

³Waterman (1970) was one of the first to face the credit assignment problem, which he encountered in the context of his program for learning Poker strategies. For a more recent attack on the credit assignment problem within game-playing programs, see Berliner (1974) and Wilkins (1980).

when one attempts to extend this method to more complex domains, variations on the basic method show promise of handling these difficulties. In conclusion, we feel that the determination of useful conditions from solution paths is an important learning technique with considerable potential, and we hope that this technique will be increasingly used in constructing intelligent systems that learn from experience.

References

- Anzai, Y. (1978) Learning strategies by computer. *Proceedings of the Second National Conference of the Canadian Society for Computational Studies of Intelligence*, 181-190.
- Berliner, H (1974) Chess as problem solving: The development of a tactics analyzer Ph D. dissertation, Computer Science Dept , Carnegie-Mellon University
- Brazdil, P. (1978) Experimental learning model. *Proceedings of the AISB Conference*, 46-50.
- Doyle, J., & London, P (1980) A selected descriptor-indexed bibliography to the literature on belief revision *SIGART Newsletter* 71:7-23.
- Langley, P (1981) Strategy acquisition governed by experimentation. CIP Working Paper No. 431, Dept. of Psychology, Carnegie-Mellon University.
- Langley, P , & Neches, R (1981) PRISM User's Manual Tech. Rep , Computer Science Dept , Carnegie-Mellon University
- Michalski, R., Carbonell, J. G , & Mitchell, T M (Eds) (in press) *Machine learning* Palo Alto, Calif : Tioga
- Mitchell, T M (1978) Version spaces: An approach to concept learning Rep. No STAN-CS-78-711, Computer Science Dept , Stanford University. (Doctoral dissertation)
- Mitchell, T M., Carbonell, J. G , & Michalski, R. (1981) Special section on machine learning. *SIGART Newsletter* 76:25-64.
- Mitchell, T. M , Utgoff, P. E., Nudel, B., & Banerji, R. (1981) Learning problem-solving heuristics through practice. *IJCAI* 7, 127-134
- Neches, R. (1981) Models of heuristic procedure modification Ph D. dissertation, Dept. of Psychology, Carnegie-Mellon University.
- Neves, D. M. (1978) A computer program that learns algebraic procedures by examining examples and working problems in a textbook *Proceedings of the Second National Conference of the Canadian Society for Computational Studies of Intelligence*, 191-195.
- Newell, A., & Simon, H. A (1976) Computer science as empirical inquiry: Symbols and search *Communications of the ACM* 19:113-126.
- Waterman, D. A. (1970) Generalization learning techniques for automating the learning of heuristics. *Artificial Intelligence* 1:121-170.
- Wilkins, D E. (1980) Causality analysis in chess. *Proceedings of the Third National Conference of the Canadian Society for Computational Studies of Intelligence*, 212-216.