December 1981

Also numbered:
HPP-81-19

Report. No. STAN-CS-81-891

The Role of the Critic in Learning Systems

by

T. G. Dietterich
B. G. Buchanan

Department of Computer Science

Stanford University
Stanford, CA 94305

The Role of the Critic in Learning Systems

T. G. Dietterich and B. G. Buchanan
Stanford University
Stanford, California 94305

Abstract

Buchanan, Mitchell, Smith, and Johnson [Buchanan 78a] described a general model of learning systems that included a component called the *Critic*. The task of the Critic was described as threefold: *evaluation* of the past actions of the performance element of the learning system, *localization* of credit and blame to particular portions of that performance element, and *recommendation* of possible improvements and modifications in the performance element. This article analyzes these three tasks in detail and surveys the methods that have been employed in existing learning systems to accomplish them. The principle method used to evaluate the performance element is to develop a *global performance standard* by (a) consulting an external source of knowledge, (b) consulting an internal source of knowledge, or (c) conducting deep search. Credit and blame have been localized by (a) asking an external knowledge source to do the localization, (b) factoring the global performance standard to produce a *local performance standard*, and (c) conducting controlled experiments on the performance element. Recommendations have been communicated to the learning element using (a) local training instances, (b) correlation coefficients, and (c) partially-instantiated schemata.

This research was supported in part by the Advanced Research Projects Agency of the US Department of Defense under contract MDA 903-80-C-0107 and by the Schlumberger-Doll Research Laboratory.

Table of Contents

List of Figures

1. Introduction

A model of learning systems has been described [Buchanan 78a] that attempts to capture the key components that must be included in any learning system. That model (shown below in Figure 1-1) is centered around the Performance Element—the component whose behavior the learning system is attempting to improve. The Performance Element (PE) responds to stimuli from the environment, and the purpose of learning is to make the responses better, in some sense. The Blackboard (BB) provides a common means of communication among the elements and ensures that all elements have access to changes made by the others. The Instance Selector (IS) selects suitable (sometimes random) training instances from the environment to present to the Performance Element. The Critic (CR) in this model evaluates the responses of the PE by comparing them against some standard of performance to determine how well the PE has done. In addition to this global evaluation, the Critic determines which parts of the PE are responsible for good and bad behavior. And, in this model, the Critic then recommends to the Learning Element (LE) what should be done to reinforce good behavior or improve bad behavior, but not precisely how to do it. Finally, the whole learning system operates within a conceptual framework, called the World Model (WM), that contains the vocabulary, assumptions, and methods that define the operation of the system.

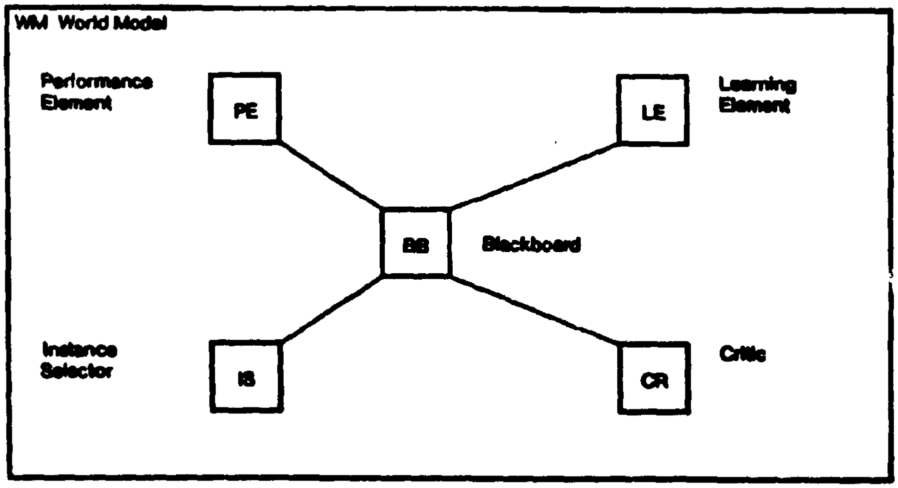

Figure 1-1: A model of learning systems (after [Buchanan 78a]).

The present paper attempts to extend this model by analyzing, in detail, the role of the Critic in existing learning systems. According the model, the Critic has three basic tasks: (a) to evaluate the current performance of the PE, (b) to localize responsibility for good and bad performance to particular parts of the PE, and (c) to make recommendations to the LE regarding desirable changes in the PE. The Critic can be viewed as an expert system that performs fault diagnosis and repair (similar to systems such as

MYCIN [Shortliffe 76] and DART [Bennett 81]). These systems evaluate the performance of some complex system (in this case, the PE), localize the causes of detected faults, and recommend repairs. The remainder of this paper discusses these three tasks in detail and then surveys the methods that have been employed in existing learning systems to accomplish them. Finally, our observations are summarized, and the implications for the design of future learning systems are assessed.

The reader is warned that many of the examples cited in this paper are necessarily brief, since the purpose of the paper is not to present existing work, but to describe and analyze the methods that have been employed to perform the Critic's three tasks. Readers desiring a fuller survey of the learning systems mentioned in this paper are encouraged to consult the article 'Learning and Inductive Inference' (Chapter XIV (in Volume 3) of the *Handbook of Artificial Intelligence* [Cohen ss]).

2. Three tasks of the Critic

2.1. Evaluation

The first and most obvious function of the Critic is to evaluate the actions of the PE. This is usually accomplished by developing a *performance standard*, that is, some sort of index against which the PE's behavior can be compared. For example, in the Meta-DENDRAL system [Buchanan 78b], the PE's task is to simulate the operation of a mass spectrometer. The simulator accepts a molecular structure as input and produces a simulated mass spectrum as output. Meta-DENDRAL's Critic employs an external performance standard in the form of an actual spectrum measured by a mass spectrometer. The subprogram INTSUM compares the simulated spectrum with the actual spectrum, and the differences serve to guide the search for new simulation rules.

The Critic, in its role as an evaluator, can be viewed in broad terms as the *test* portion of a generate-and-test method. The learning element is the generator. It proposes modifications in the PE, and the Critic tests these modifications by evaluating the actions of the PE on particular training instances. Every learning system can thus be viewed as learning by trial and error (usually heuristically-guided trial and error). It is through the PE and the Critic that the hypotheses developed by the LE are tested empirically.

2.2. Localization of responsibility

The second task of the Critic—to localize responsibility for good and bad behavior to particular portions of the PE—was first pointed out by Minsky [Minsky 63], where he called it the *credit-assignment problem*. The credit-assignment problem arises whenever the PE has *composite structure* and only a *global performance standard* is available. By composite structure, we mean that the decisions of the PE are determined by some composite decision-making process. The PE, for example, may evaluate a complex expression or apply a series of rules to arrive at a decision. In order for the LE to improve individual subexpressions or individual rules, the Critic must localize credit and blame for overall performance to these particular subexpressions or rules.

For example, consider the learning problem addressed by Mitchell's LEX system [Mitchell 81]. LEX solves symbolic integration problems. This performance task has composite structure: In order to solve an integral, LEX must apply a *sequence* of integration operators. Furthermore, LEX has only a global performance standard; it knows it has solved the problem when it has succeeded in removing the integral sign from the expression being integrated. Once LEX has found a sequence of operators that solves the problem, it must apportion credit and blame among the individual operators in that sequence and among operators in any other unsuccessful sequences that it investigated.

The credit-assignment process can be viewed as the process of converting a *global performance standard* into a *local performance standard*. The local performance standard indicates what the proper outcome of each move (or each subdecision) should have been. Once a local standard is obtained, credit assignment is straightforward.

LEX must break its global performance standard, which indicates how good an entire solution path is, into a local performance standard that indicates how good each step (each application of an integration operator) is. The exact global performance standard used by LEX is the length of the shortest known solution path, as measured by computation time and space. The local performance standard for each solution step is the length of the shortest known path from the starting state of that step to a solution state. These various performance standards are shown in Figure 2-1.

Once LEX has developed the local performance standard for each step, it can complete the credit-assignment process by evaluating every step in its tree of solution paths and partial paths. A step is judged to be a good decision if it leads to a solution whose path length is less than 1.15 times the local performance standard for that step. Otherwise, the step is judged to be a bad step. This has the effect of crediting all steps on the best known path (and on any other paths that have nearly the same length as the best known path). Blame is assigned to any step that leads from a state on the shortest path to a state not on the shortest path. All other decisions remain unevaluated (see Figure 2-2).

It is important to observe that the credit-assignment problem only arises when the units of knowledge being learned constitute small subcomponents of the PE. There would be no credit-assignment problem in LEX if, instead of trying to learn heuristics for individual integration operators, LEX simply memorized

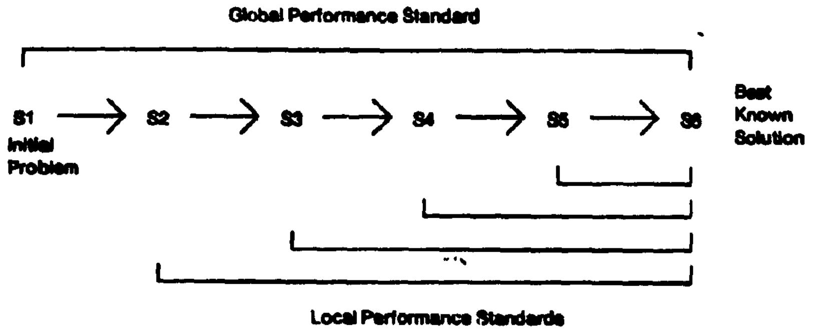

Figure 2-1: Global and local performance standards in LEX.

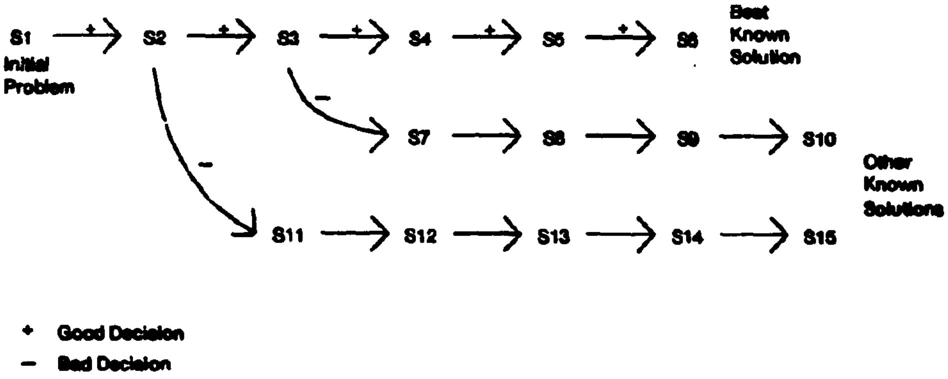

Figure 2-2: Credit assignment in LEX.

complete operator sequences. Exactly this kind of learning is performed by STRIPS [Fikes 72]. Furthermore, one can imagine creating a learning system in which a series of credit-assignment problems must be solved. Such a system could have, for example, a layered performance element in which the overall decision was made by a set of production rules, and each production rule depended in turn on evaluating some complex expression. The first credit-assignment problem would be to develop a local performance standard for each production rule. This would be followed by a second credit-assignment process in which a local performance standard would be computed for each subexpression that makes up each production rule. In summary, the difficulty of the credit-assignment problem depends on the level of analysis carried out by the LE. As the relative size of the "learnable unit" of knowledge becomes smaller, the credit-assignment problem becomes more difficult.

2.3. Recommending changes in the ?erformance Element

Once global and local performance standards have been obtained and, thus, the causes of poor performance have been isolated, the Critic must recommend to the LE how the PE should be modified. These recommendations can be thought of as verbs such as *generalize*, *specialize*, and *replace*, along with some information that indicates what should be modified.

In systems that learn from examples, the recommendation is usually to generalize or specialize a particular rule or concept in order to make it consistent with some new training instances. In LEX, for example, the final output of the Critic is a set of instances of the proper (and improper) application of integration operators, gleaned from the trace of the problem solving process. These training instances are supplied to the learning element along with instructions to generalize or specialize the heuristics that recommended the use of those integration operators.

In other learning systems, the recommendations may take the form of fairly specific instructions for how to modify the knowledge base. In Sussman's HACKER [Sussman 75], for example, the Critic provides a partially filled-in schema describing a Conniver demon (or, more correctly, an "if-added method"). The learning element must fully instantiate this demon and install it in the knowledge base. The schema includes instructions for how to generalize certain parts of the demon, as well.

The dividing line between the Critic and the LE is not always clear. In many learning systems, there is no separate recommendation phase. Instead, the LE directly employs the local performance standard to modify the PE. In systems that discover single concepts from training instances (such as Winston [Winston 70], Mitchell [Mitchell 78], Michalski [Michalski 78], and Hayes-Roth [Hayes-Roth 78]), the information provided by the training instances and their correct classifications suffices to guide the LE. Mitchell's version space algorithm [Mitchell 78], for example, applies a matching process directly to the training instances themselves in order to decide how the current concept description should be modified. The performance standard determines the nature of the change: Positive training instances lead to generalization, and negative instances lead to specialization, of the concept description.

3. Methods for obtaining a Global Performance Standard

Now that we have reviewed each of the three tasks of the critic, we turn our attention to the methods that have been used in existing systems to accomplish these tasks. Several methods have been employed for finding a global performance standard. These can be grouped into three general categories: knowledge sources external to the Critic, knowledge sources within the Critic, and search.

3.1. Knowledge sources external to the Critic

The first method of obtaining the global performance standard is to ask the outside world to provide one. We have already mentioned Meta-DENDRAL's use of an actual mass spectrum as a standard of performance for its mass spectrometer simulator. Many programs that learn concepts from examples expect the training instances to be correctly classified in the input. Winston's [Winston 70] ARCH learning system, for example, relies on the teacher to indicate for each training instance whether that instance is an "arch" or a "near miss."

Another system that employs an external performance standard is Samuel's checkers program [Samuel 63, Samuel 67]. One configuration of the checkers program uses an outside knowledge source in the form of "book moves"—moves taken from recorded checkers matches between masters. Samuel's program attempts to learn an evaluation function that computes the worth of a given board position. The program learns by following book games, first applying its current evaluation function in order to select a move and then comparing the selected move with the global performance standard—the book move.

Davis' TEIRESIAS program [Davis 76] provides another example of a system that turns to an expert for performance feedback. TEIRESIAS provides knowledge acquisition and debugging support for EMYCIN-based expert systems. The EMYCIN system serves as the performance element. It is presented with cases, which it processes by applying the rules in its knowledge base. When the consultation is completed, TEIRESIAS steps in and asks the expert whether the PE's conclusions are correct. At this point, the expert responds with a simple YES or NO. If the answer is NO, TEIRESIAS assists the expert in actually locating and repairing the problem (typically a missing or incorrect rule). The expert's yes/no answer serves as the global performance standard.

The technique of obtaining the global performance standard by consulting some source of knowledge external to the program is most useful in situations where the PE is attempting to model or mimic the behavior of a physical system or a human expert. In such cases, if the physical system is opaque or the human expert is unable to introspect well, the only information available to the learning system is the overall behavior of the unknown system. This global behavior can serve as the global performance standard.

3.2. Knowledge sources internal to the Critic

A second approach to obtaining a global performance standard is to employ some source of knowledge inside the Critic. Waterman's poker player [Waterman 70], in its implicit training mode, is a good example of such a system. The performance task of the poker player is to decide what bets to make during a round of play of draw poker. This is a composite task, since a *sequence* of decisions must be made. At the end of each round of play, Waterman's Critic invokes an internal knowledge source to produce the global performance standard. The knowledge source is a rule-based system containing an axiomatization of the rules of draw poker along with rules describing how bets accumulate and how betting behavior is related to the quality of the players' hands. It contains definitions of the four basic actions available to the PE (CALL, DROP, BET HIGH, and BET LOW). The rule describing the CALL action, for example, is represented as

```
        ACTION(CALL) & HIGHER(YOURHAND, OPPHAND) =>
                       ADD(LASTBET, POT) & ADD(POT, YOURSCORE).
```

(I.e., if you call and your hand is superior, then you win the pot as augmented by your last bet.)

To evaluate a round of play, the Critic first determines the truth values of certain predicates such as GOOD(OPPHAND) and HIGHER(OPPHAND, YOURHAND) and then tries to prove the statement MAXIMIZE(YOURSCORE) by backward chaining through the rule base. The resulting proof indicates whether the PE could have won more money than it did.

Internal sources of knowledge are useful in domains where it is possible to encode some—but not all—of the knowledge needed to guide the performance element. For poker, it is easy to provide the basic rules of the game to the program. Unfortunately, the PE needs to know more than just the rules in order to play well. Consequently, the only use of the poker rule base in Waterman's system is to provide the Critic with a global performance standard.

An interesting characteristic of poker—and of many other task domains such as medicine, law, and politics—is that expertise consists of knowing *in advance* what actions should be taken. It is relatively easy to tell retrospectively what the performance element should have done. In these task domains, if the knowledge required for retrospective analysis can be incorporated into the learning system, then it can provide a global performance standard for the PE.

3.3. Search

In most *problem solving* tasks, deeper searches provide more information about the best solution. In LEX, for example, deeper and wider search leads to several alternative solutions to the integral. As we have seen above, LEX uses this fact to obtain the global performance standard. LEX chooses the path length of the shortest known solution and uses it as an upper bound on the lengths of other paths. During problem solving and credit assignment, whenever a path exceeds the upper bound, it is dropped from further consideration.

Samuel's checkers player—in an alternate configuration that does not employ "book moves"—uses deep search to obtain its global performance standard. Recall that Samuel's system is attempting to learn an evaluation function for board positions. One way to determine the quality of a board position is to search deeper into future game positions, apply the same evaluation function to the tip positions, and compute the mini-max backed-up value. Since the backed-up value based on a deep search is more accurate than the value calculated directly from the board position in question, it can serve as a global performance standard for the evaluation function.

In summary then, there are three basic approaches to finding a global performance standard. In domains where a physical system or an expert is being modeled, the external environment can provide the performance standard in the form of the actual behavior of the physical system or the expert. In domains where retrospective analysis is easy, the knowledge required for such retrospective analysis can provide the standard. Finally, in domains where deeper searches produce more information, simple search can provide a global performance standard.

4. Methods for assigning credit and blame—obtaining a Local Performance Standard

Once we have an overall performance standard, how can we localize credit and blame to individual decisions? Three basic methods can be discerned. One approach is to side step the problem by consulting some knowledge source outside the program, a second approach is to factor the global performance standard into a local performance standard, and the third approach is to conduct controlled experiments by varying some subcomponent of the PE and observing the resulting changes in the global behavior of the PE.

4.1. Knowledge sources external to the Critic

Of course it is possible to finesse the credit-assignment problem completely by simply asking the external world to provide a move-by-move performance standard. In one configuration of Waterman's poker learner, for example, a human expert provides feedback after each bet decision. TEIRESIAS also relies on the human expert to examine the performance trace and localize the point at which the PE went wrong. This approach is useful in situations where an expert is available who can successfully criticize particular cases. The particular cases serve to focus the expert's attention and trigger his or her memory. This is an important aspect of the standard knowledge engineering methodology [Davis 76].

Another situation in which the external world provides the local performance standard is program debugging. When a programmer is testing a program (i.e., a PE), he or she must have some idea what the proper outputs of the program should be—that is, the programmer must know the global performance standard. When one of the outputs is incorrect, interactive debugging tools, such as the INTERLISP BREAK package, enable the programmer to inspect intermediate states within the program. However, the programmer must also have some idea what the correct internal states should be—that is, the programmer must figure out what the local performance standard is. Some programming language features, such as run-time type checking and run-time correctness assertions, allow the programmer to partially specify the local performance standard so that the programming system can automatically compare it with the actual behavior of the program.

4.2. Factoring the global performance standard

A second approach to solving the credit-assignment problem is to factor the global performance standard into local standards that correspond to the subparts or subdecisions of the PE. This approach relies on discovering some substructure within the global performance standard. In Meta-DENDRAL, for example, the global performance standard—the actual mass spectrum for a molecule—is factored into its individual lines. This factoring takes advantage of the fact that the global performance standard has some substructure—it is made up of spectral lines. Of course not *any* factorization will work. The spectral lines must correspond somehow to the PE subcomponents that the Critic is attempting to evaluate. In Meta-DENDRAL, the subcomponents of interest within the PE are cleavage rules that predict, for a given molecular bond cr set of bonds, whether those bonds will break. Each spectral line corresponds to some combination of one to three individual cleavages.

In Meta-DENDRAL, the credit-assignment process is carried out by the subprogram INTSUM, which is a transparent version of the PE (the mass spectrometer simulator). When INTSUM is given a molecule, it simulates the cleavage process and produces a simulated spectrum. More importantly, however, each line in the simulated spectrum is annotated with a record of which cleavages led to the creation of that line. Thus, the correspondence between spectral lines and PE subcomponents is computed. Now credit assignment is trivial. If the simulated line matches an actual line, then the cleavages that "caused" the simulated line are credited. Otherwise, if the simulated line does not match a real line, the cleavages are blamed.

Meta-DENDRAL starts the learning process with a "half-order theory" of the mass spectrometer. This half-order theory can be thought of as a small set of very general cleavage rules that state that just about every bond in the molecule will break, with a few exceptions (e.g., double bonds, triple bonds, bonds in aromatic rings, and bonds incident to the same atom). The learning process improves the half-order theory by specializing it to predict more precisely when bonds will break.

Figure 4-1 provides a schematic diagram of the process of factoring the global performance standard, comparing it to the intermediate decisions of the PE, and assigning credit and blame to various subcomponents of the PE.

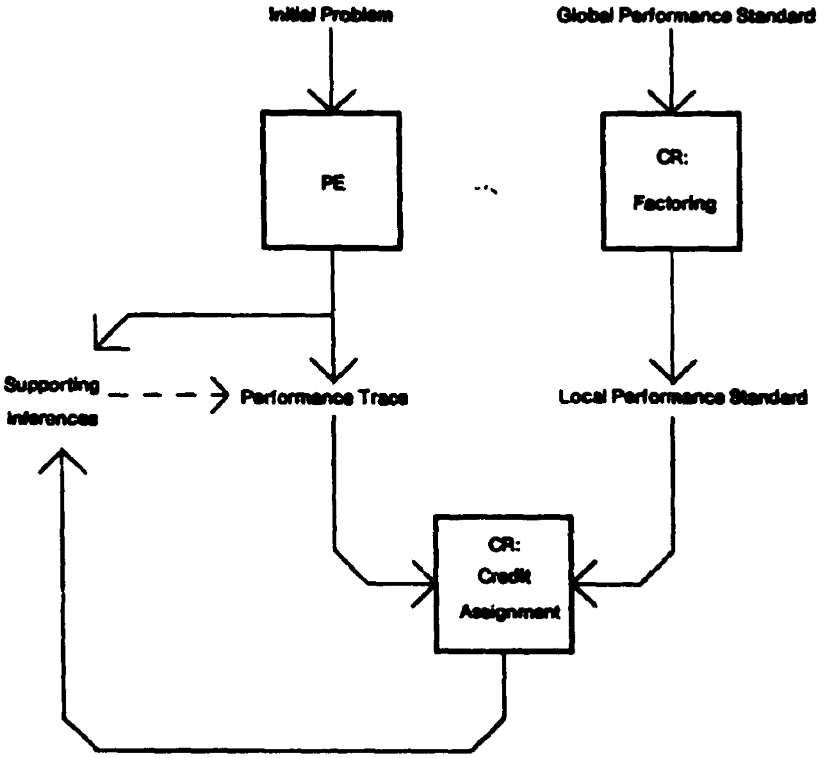

Figure 4-1: Diagram of the factoring and credit-assignment process.

A second system that successfully factors the global performance standard is Sussman's HACKER system [Sussman 75]. HACKER is a blocks-world planner; given an initial blocks configuration and a desired configuration, it must develop a sequence of operations (a plan) that will achieve the given goal. HACKER employs an internal knowledge source—a blocks world simulator—as its global performance standard. Once the PE (the planner) has developed a plan, the simulator simulates it to see if the plan will in fact attain the goal.

The simulator can be factored to obtain a local performance standard. The simulation is conducted one step at a time; after each step, the state of the world can be compared with what the PE expected it to be. In fact, the simulator detects three kinds of errors: illegal actions, violated expectations, and unaesthetic actions. Illegal actions are actions, such as picking up a whole stack of blocks, that are illegal in the blocks world. Violated expectations are precisely that—steps whose intended effects were not achieved. Unaesthetic actions are actions in which the program moves the same block two times in succession with no intervening action. Once one of these errors is detected, HACKER proceeds immediately to develop a bug demon that will detect the problem and patch around it in future plans. HACKER does not conduct further credit-assignment to determine which of its planning methods was at fault. Planning methods—such as the method that states that conjunctive goals can be achieved independently—are never modified; badly formed plans are just patched prior to execution.

In order for HACKER's credit-assignment strategy to work, it is very important that the PE provide a detailed trace of its planning process. This trace lists the subgoals that each plan step is expected to achieve. The simulator compares these expectations with the simulated execution of the plan and localizes blame accordingly.

Waterman's poker player [Waterman 70] is a third system that factors the global performance standard. Recall that the Critic attempts to prove the statement MAXIMIZE(YOURSCORE) using an axiom system that encodes the rules of poker and some knowledge about how bets accumulate. The proof provides a global performance standard; it indicates whether or not the performance element could have improved its winnings during the round of play. In addition, the proof provides a local performance standard. It discovers the sequence of bet decisions that would have led to the best score for the program.

The description of the axiom system given above is slightly inaccurate. It gives the impression that only one proof is conducted for each round of play. In fact, a separate proof of the MAXIMIZE(YOURSCORE) statement is conducted for *each* bet decision in the round. Waterman has analyzed, in advance, all of the ways in which previous bet decisions can influence subsequent bet decisions. His analysis is incorporated into the axiom system using a few predicates such as LASTBETOPP(BET HIGH), which says that the opponent's last bet was high. For each bet decision, the truth values of such "connective" predicates are determined by examining previous bets, and then the axiom system is invoked to see if the current bet was appropriate. In essence, Waterman has manually factored the performance standard so that it can be applied to individual bet decisions.

The three systems just described—Meta-DENDRAL, HACKER, and Waterman's poker player—all obtain their local performance standards by factoring the global performance standard. Meta-DENDRAL factors the spectrum into its individual lines, HACKER factors the overall simulation of the plan's execution into the simulation of each plan step, and Waterman factors the proof for the whole round of play into individual proofs for each bet decision.

4.3. Conducting controlled experiments

The third approach to solving the credit-assignment problem is to modify some subcomponent of the PE and observe how the global performance changes. This is the technique employed by computer engineers when they attempt to localize a fault by swapping a single printed-circuit board and then observing the overall behavior of the system to see if the problem goes away. It is a powerful technique, but it only provides unambiguous information if the global performance of the PE actually changes. If changing a subcomponent has no effect, it is difficult to distinguish the case in which the subcomponent is unimportant from the case in which the component is vital, but a second problem in the PE is masking the effects of the component change.

Three existing learning systems can be viewed as performing controlled experiments in order to localize PE faults. One system is Samuel's checkers player, which attempts to learn the coefficients of a polynomial evaluation function. In order to assign credit and blame to individual coefficients, Samuel computes the pairwise correlation between the value of each checkers board feature and the global performance standard. Features whose changes correlate positively with the global performance standard are given positive coefficients, and features that vary inversely with the global performance standard are given negative coefficients.

This approach to solving the credit-assignment problem makes an independence assumption: It assumes that credit and blame can be allocated to each part of the PE independently. For the polynomial evaluation function, this makes sense because the use of the polynomial itself is based on the premise that the overall value of a checkers move can be obtained by computing a weighted sum of various board features. Samuel's research has shown, however, that the linear polynomial fails to capture much of the knowledge employed by checkers masters because of this independence assumption. Hence, the pairwise correlation approach, although adequate for the polynomial representation, may not be adequate in general.

The second system that employs controlled experiments is Samuel's signature table checkers system. This system uses a signature table, instead of a polynomial, to represent the evaluation function. A signature table is an n-dimensional array of cells. Each dimension corresponds to some numerical checkers board feature, such as the number of kings or the number of open squares in the back row. The features are measured and then used to index into the signature table to obtain the corresponding cell. The cell contains a numerical rating of that board position. Thus, in principle, the signature table can represent a different rating for each distinct board position, and hence, the independence assumption is not needed. Samuel's experiments with signature tables indicated that they did indeed perform better than simple polynomial evaluation functions for this reason.

Unfortunately, a single signature table for Samuel's 24 checkers board features would be prohibitively large (roughly 10^{12} cells). Instead, Samuel employs a three-level tree of smaller tables as shown in Figure 4-2. The PE first determines the values of the 24 board features. These values are used to index into the first set of signature tables to obtain new values. Each of these first-level values is then used to index into the second-level tables where the cells contain values that in turn serve as indexes for the final third-level table. The cells in the third-level table provide the actual evaluation of the board position.

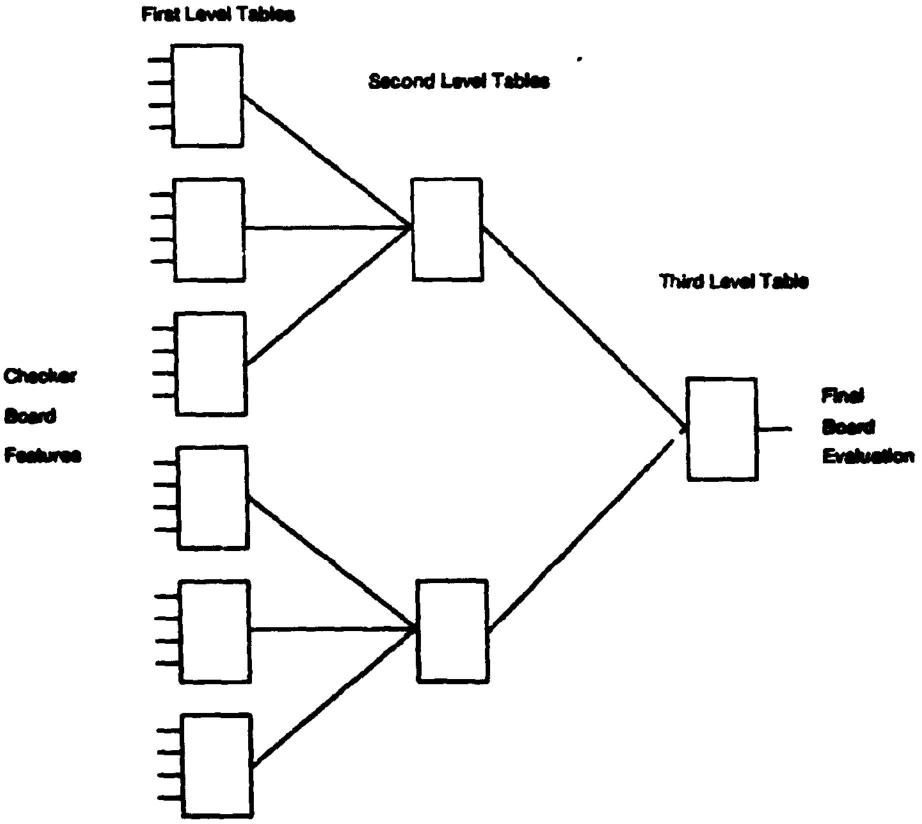

Figure 4-2: Samuel's three-level signature table scheme.

We can think of this three-level set of tables as a three-level production system in which the results of the first-level inferences are tested in the condition parts of the second-level production rules. The second-level rules produce values that serve to trigger third-level rules. Credit assignment for signature tables is accomplished by computing a correlation coefficient for each cell in each table, in a fashion similar to that used for the polynomial evaluation functi›n. Each cell is associated with two tallies, called A and D, which are initially zero. At each board position during a game, the program is faced with a set of alternative moves, one of which is indicated by the performance standard to be the best move. Each possib'e move can be mapped to one cell in *each* signature table. A 1 is added to the D tally of each cell whose corresponding move is *not* the correct move, and a total of *n* (where *n* is the number of incorrect alternatives) is added to the A total of each cell correponding to the correct move. From the A and D totals, a correlation coefficient C = (A − D)/(A + D) is computed and used to update the contents of the cells in the signature tables.

This tally method effectively credits the entire inference tree corresponding to the correct move and blames all alternative trees. This is not a problem for the first-level cells, since it is always clear which cells correspond to the best board position. However, for the second- and third-level tables, the cells corresponding to the best move are not necessarily those cells selected by the first-level tables—especially during the early phases of learning when the first-level tables still contain incorrect values. Samuel's system credits them anyway and relies on a huge number of training instances (approximately 250,000 moves) to correct eventually any errors introduced by this procedure. Samuel also found it necessary to apply interpolation procedures to the signature tables during the early phases of learning, since, at that point, most cells in the first-level tables were empty. Thus, the pairwise correlation method of assigning credit and blame is not entirely adequate for signature tables either, but the large number of training instances gradually corrects any errors introduced through this credit-assignment process.

Finally, the third system that conducts controlled experiments is LEX. As we have seen above, LEX evaluates alternative operator applications by investigating the subtrees rooted at each of the alternatives. The move that leads to the shortest subtree is credited, and all moves leading to significantly larger subtrees are blamed. This method works well as long as the subtrees are carefully investigated. Unfortunately, during the early stages of learning, the performance element is easily overwhelmed by combinatorial explosion and, hence, cannot fully investigate these subtrees. Even more troublesome is the behavior of the performance element when it has learned an overly specific heuristic. Such a heuristic causes it to ignore an operator even when that operator should be applied. This leads the PE to overlook possible solutions and hence, resulting in incomplete investigation of the alternative subtrees.

We have examined three existing systems that employ some form of controlled experimentation to localize credit and blame: Samuel's polynomial system, Samuel's signature table system, and Mitchell's LEX system. In general, controlled experimentation—that is, systematic variation of a single subcomponent of the PE—is a powerful method. There are some difficulties however. First, it is necessary to assume some independence among the faults within the PE. If multiple interdependent problems exist, then one fault can mask another and varying a single component will not necessarily lead to any change in system performance.

A second difficulty is that, in some performance elements, adequate controls may not be available. In order to create a controlled experiment, it must be possible to vary some aspect of the performance element while keeping all other parts fixed. This is not possible in LEX, for example, where choosing a different integration operator leads to a different search tree, which must be thoroughly investigated in order to permit accurate comparisons. This is a kind of factoring of the performance element into nearly independent subcomponents. Such factoring is similar to the factoring of the global performance standard.

A question for future research is to try to understand exactly when the PE or the global performance standard can be properly factored and to develop factorization methods. This is equivalent to the deeper question of what structures are ultimately learnable. Do there exist systems that are so complex that they cannot be sufficiently factored to allow learning to occur? One speculation is that a task that cannot be factored can still be learned at a more abstract level of analysis. Suppose, for example, that adequate controls were not available in LEX to evaluate individual applications of integration operators. It would still be possible to expand the level of analysis and simply memorize entire sequences of operators. In tightly coupled interdependent systems, there may be no useful detailed level of analysis. The heuristics describing when a particular operator should be applied would necessarily become extremely complex and lengthy in order to capture all of the interdependencies in such a system[1]. Thus, the credit-assignment problem may be solvable in all cases either by successfully factoring the global performance standard, by factoring the PE and conducting controlled experiments, or by changing the level of analysis to learn larger units of knowledge.

5. Methods for developing recommendations

Once the problems have been localized to individual decisions within the PE, the PE must be modified so that these problems do not recur. There are many ways that a performance element can be repaired. In production systems, for example, the antecedents of the production rules can be generalized or specialized, the consequents of the rules can be altered to perform different actions, new rules can be added, and conflict-resolution strategies can be modified. In concept-learning systems, the definition of the concept can be generalized, specialized, or completely changed. Sometimes a decision must be made about whether to repair the problem directly or create a demon to detect the problem in the future and patch around it.

The final task of the Critic is to figure out what *kind* of change should be made, rather than to actually make the modifications itself. In order to accomplish this, the Critic must have some way of mapping kinds of faults into kinds of repair. This mapping has been fairly straightforward in existing learning systems. In LEX,

[1] Notice, however, that this argument does not address the possibility that the entire task could be reformulated to reveal a much simpler internal structure (e.g., as Ptolemaic astronomy was superseded by Copernican astronomy).

for example, if an integration operator was applied in a situation when it should not have been, then the solution is to specialize the heuristic attached to that operator so that it will no longer recommend its use in such situations. Conversely, if an integration operator should have been applied, but was not, then the solution is to generalize the associated heuristic.

For HACKER, the mapping between kinds of faults and proposed fixes is more indirect. This is because the basic cause of all of HACKER's errors is its "linearity assumption"—that conjunctive goals can be pursued completely independently. Rather than modifying this assumption, HACKER's solution is to patch around it. Consequently, when a fault is detected, HACKER must decide which particular kind of patch should be applied. This is accomplished by examining the subgoal structure of the failed plan step and matching this structure against a set of subgoal interaction schemata. The matching schema has an associated skeletal demon that, when fully instantiated and installed in the knowledge base, will detect future instances of the problem and fix them prior to the execution of the plan.

In addition to indicating what kind of repair should be undertaken by the learning element, the Critic usually provides the LE with additional information to guide the modifications. In systems that learn from examples, the Critic provides *local training instances* to the LE. A local training instance is an instance of the use of a particular rule along with the local performance standard for that rule application. This is to be distinguished from a global training instance, which has the form of a global problem situation paired with a global performance standard. In Meta-DENDRAL, for example, a global training instance has the form of a known molecular structure and its associated actual mass spectrum. The local training instances are comprised of individual bond cleavages and the corresponding spectral lines that they produce. It is these local training instances that are generalized by the LE to develop general cleavage rules.

In LEX, the global training instance is the initial integration problem coupled with the global performance standard (i.e., the shortest known solution path length). The local training instances are the positive (or negative) examples of the correct (or incorrect) application of integration rules. These local instances are processed by the candidate-elimination algorithm [Mitchell 78] to develop integration heuristics.

Instead of local training instances, Samuel's checkers Critic provides a set of correlation coefficients to the LE. As we saw above, the correlation coefficients indicate how the individual features in the evaluation function are correlated with the global performance standard. These correlation coefficients are rescaled by the LE and then substituted for the previous weights in the evaluation polynomial.

Finally, as we mentioned above, HACKER's Critic provides the LE with a partially-filled-in bug demon that will detect and correct bugs in future plans. The skeletal demon provides instructions for how to complete the instantiation process and tells which parts of the demon should be generalized.

In summary, the task of deciding what kind of changes should be recommended to the LE is fairly straightforward. In existing systems, the recommendations have been communicated to the LE using local training instances, correlation coefficients, and skeletal demons.

6. Summary

In this paper, we have reviewed the three tasks of the Critic: obtaining a global performance standard, converting it into a local performance standard, and recommending to the learning element how the PE should be improved. For each of these tasks, we have attempted to list the methods that have been employed in existing learning systems to accomplish them. Global performance standards have been obtained by asking the external environment to supply them, by consulting some internal source of knowledge, and by conducting deeper searches. Local performance standards have been obtained by asking external sources, by factoring the global performance standard in some way, and by conducting controlled experiments. Finally, recommended changes have been communicated to the LE in the form of verbs such as *generalize*, *specialize*, or *replace* along with information such as local training instances, correlation coefficients, and skeletal bug demons.

One of the most interesting results of this analysis is the conclusion that the credit-assignment problem may be solvable in most learning situations. If a system can be factored, then controlled experiments can be conducted to localize faults. If a system resists factoring, then the level of analysis can, and should, be modified to learn larger units of knowledge. A disadvantage of learning these larger units of knowledge is that many more of them will need to be learned. However, in complex systems it is likely that nc simple generalizations can be found at the level of individual rules, and hence, the learning system has no choice but to learn larger units of knowledge.

Another result of this analysis is the observation that the Critic requires a model of the internal operation of the PE and some understanding of the semantics of the PE's knowledge base. Credit and blame cannot be localized to individual rules in the knowledge base unless the Critic has access to and can understand those rules. Similarly, recommendations for change require at least some understanding of how the PE interprets the knowledge base.

Thirdly, the Critic must have access to a trace of the internal decision-making process of the PE. Without this additional information, it is impossible to compare the PE's internal decisions with the local performance standard.

Finally, this paper has analyzed the Critic as if it were a separate expert system with its own knowledge base. We have attempted to catalog the kinds of knowledge that such an expert system would need and the kinds of methods it would apply. However, most of this knowledge is still not well formulated. Making it explicit is a necessary part of the research needed to provide knowledge acquisition tools for expert systems.

7. Acknowledgments

The authors wish to thank the Advanced Research Projects Agency of the US Department of Defense and the Slumberger-Doll Research Laboratory for supporting this research. Thanks also go to Charles P. Paulson and James S. Bennett for comments on early drafts of this paper.

References

[Bennett 81] Bennett, J. S., Hollander, C. R.
DART: An expert system for computer fault diagnosis.
In *Proceedings of IJCAI-81*. Vancouver, Canada, August, 1981.

[Buchanan 78a] Buchanan, B. G., Mitchell, T. M., Smith, R. G. and Johnson, C. R. Jr.
Models of learning systems.
Encyclopedia of Computer Science and Technology 11, 1978.
Also Stanford report STAN-CS-79-692.

[Buchanan 78b] Buchanan, B. G. and Mitchell, T. M.
Model-directed learning of production rules.
In Waterman, D. A. and Hayes-Roth, F. (editors), *Pattern-Directed Inference Systems*, .
Academic Press, New York, 1978.

[Cohen ss] Cohen, P. and Feigenbaum, E. A.
The Handbook of Artificial Intelligence.
Tioga, Palo Alto, CA, In press.

[Davis 76] Davis, R.
Applications of meta-level knowledge to the construction, maintainance, and use of large knowledge bases.
Technical Report STAN CS-76-552, Stanford University, June, 1976.

[Fikes 72] Fikes, R. E., Hart, P. E. and Nilsson, N. J.
Learning and executing generalized robot plans.
Artificial Intelligence 3:251-288, 1972.

[Hayes-Roth 78] Hayes-Roth, F. and McDermott, J.
An Interference Matching Technique for Inducing Abstractions.
Communications of the ACM 21(5):401-410, 1978.

[Michalski 78] Michalski, R. S.
Pattern recognition as knowledge-guided induction.
Technical Report Report #927, Department of Computer Science, University of Illinois at Urbana, 1978.

[Minsky 63] Minsky, M.
Steps Toward Artificial Intelligence.
In Feigenbaum, E. A. and Feldman, J. (editors), *Computers and Thought*, pages 406-450.
McGraw-Hill, New York, 1963.

[Mitchell 78] Mitchell, T. M.
Version Spaces: An approach to concept learning.
PhD thesis, Stanford University, December, 1978.
also Stanford CS report STAN-CS-78-711, HPP-79-2.

[Mitchell 81] Mitchell, T. M., Utgoff, P. E., Nudel, B., and Banerji, R. B.
Learning Problem-Solving Heuristics Through Practice.
In *Proceedings of IJCAI-81*. Vancouver, Canada, August, 1981.

[Samuel 63] Samuel, A. L.
Some studies in machine learning using the game of checkers.
In Feigenbaum, E. A. and Feldman, J. (editors), *Computers and Thought*, pages 71-105.
McGraw-Hill, New York, 1963.

[Samuel 67] Samuel, A. L.
Some studies in machine learning using the game of checkers II - recent progress.
IBM Journal of Research and Development 11(6):601-617, 1967.

[Shortliffe 76] Shortliffe, E. H.
Computer Based Medical Consultations: MYCIN.
American Elsevier, New York, 1976.

[Sussman 75] Sussman, G. J.
A Computer Model of Skill Acquisition.
American Elsevier, New York, 1975.

[Waterman 70] Waterman, D. A.
Generalization learning techniques for automating the learning of heuristics.
Artificial Intelligence 1(1/2):121-170, 1970.

[Winston 70] Winston, P. H.
Learning structural descriptions from examples.
Technical Report MIT AI-TR-231, MIT, Cambridge, Mass., September, 1970.