May 1988 UILU-ENG-88-2221

COORDINATED SCIENCE LABORATORY
College of Engineering

TOWARD INTELLIGENT MACHINE LEARNING ALGORITHMS

Robert E. Stepp
Bradley L. Whitehall
Lawrence B. Holder

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-88-2221	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois	6b. OFFICE SYMBOL (If applicable) N/A
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Avenue Urbana, IL 61801	7a. NAME OF MONITORING ORGANIZATION NSF, ONR, DARPA
	7b. ADDRESS (City, State, and ZIP Code) 1800 G. Street, Washington D.C. 20552 800 N. Quincy, Arlington VA 22217 1400 Wilson Blvd., Arlington, VA 22209-2308
8a. NAME OF FUNDING/SPONSORING ORGANIZATION NSF, ONR, DARPA	8b. OFFICE SYMBOL (If applicable)
	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER NSF IST - 85-11170 N00014-82-K-0186, N00014-87-K-0874
8c. ADDRESS (City, State, and ZIP Code) 1800 G. Street, Washington, D.C. 20552 800 N. Quincy, Arlington VA 22217 1400 Wilson Blvd, Arlington VA 22209-2308	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Toward Intelligent Machine Learning Algorithms

12. PERSONAL AUTHOR(S) Stepp, Robert E., Whitehall, Bradley L., Holder, Lawrence B.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	May 1988	16

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Machine learning knowledge-directed systems Discovery

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Machine learning is recognized as a tool for improving the performance of many kinds of systems, yet most machine learning systems themselves are not well equipped to improve their own learning performance. By emphasizing the role of domain knowledge, learning systems can be crafted as knowledge-directed systems, and with the addition of a knowledge store for organizing and maintaining knowledge to assist learning, a *learning* machine learning (*L*-ML) algorithm is possible. The necessary components of *L*-ML systems are presented along with several case descriptions of existing machine learning systems that possess limited *L*-ML capabilities.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

Towards Intelligent Machine Learning Algorithms*

Robert E. Stepp
Bradley L. Whitehall
Lawrence B. Holder

Artificial Intelligence Research Group
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1101 West Springfield Avenue
Urbana, IL 61801

Telephone: (217) 333-6071
Internet: stepp@uicsl.csl.uiuc.edu

March 1988

ABSTRACT

Machine learning is recognized as a tool for improving the performance of many kinds of systems, yet most machine learning systems themselves are not well equipped to improve their own learning performance. By emphasizing the role of domain knowledge, learning systems can be crafted as knowledge-directed systems, and with the addition of a knowledge store for organizing and maintaining knowledge to assist learning, a *learning* machine learning (L-ML) algorithm is possible. The necessary components of L-ML systems are presented along with several case descriptions of existing machine learning systems that possess limited L-ML capabilities.

* This research was partially supported by the National Science Foundation under grant NSF IST-85-11170, the Office of Naval Research under grant N00014-82-K-0186, by the Defense Advanced Research Projects Agency under grant N00014-87-K-0874, and by a gift from Texas Instruments, Inc.

1. INTRODUCTION

Machine learning purports to be an important tool for increasing the performance of knowledge-intensive problem solving tasks. The ability to learn has been identified as a primary ingredient in any intelligent system; there is even hope that machine learning systems can conquer the knowledge-acquisition bottleneck and ultimately provide knowledge that can surprise their human creators [Michalski86]. It seems ironic that relatively little attention has been given to machine learning itself as a knowledge-intensive problem solving task—a task whose performance we would like to see improve with experience and with the availability of more problem dependent and problem independent knowledge. Learning itself should be conducted in an intelligent way, especially because learning is hard.

By its nature, it is unlikely that any single conceptualization of a learning algorithm will be satisfactory for even a modest range of learning situations. An intelligent adaptive learning algorithm is surely necessary. The focus of this paper is on machine learning algorithms that can learn, i.e., that can change their own performance as they gain experience.

Today's typical machine learning algorithm does not improve its own performance over time, but remains static. When faced with another learning problem, even one identical to a problem seen before, the same computations are performed again, taking no advantage of biases or constructions or generalizing transformations that have already been shown to be effective for that class of circumstances. The same problem solution space is explored again, as if it were fresh and previously unexplored. Clearly a human demonstrating such behavior would not be called intelligent.

There are several key ingredients required to make a *Learning* Machine Learning (*L*-ML) System. Principally, an *L*-ML system must both be directed by

and be an updater of background knowledge. Specifically, an *L*-ML system needs

- the ability to use background knowledge to transform representation spaces[1],
- the ability to direct learning under the influence of a specified goal in the context of a goal hierarchy[2],
- the ability to compose generalizations and simplifications from one or more concepts in background knowledge, efficiently reusing acquired generalizing and structuring concepts,
- the ability to discover patterns in examples, background knowledge, biases, and goals that are effective for learning,
- the ability to update a background knowledge store with discovered general, domain-specific, and problem-specific characteristics paired with the control knowledge that was used to accomplish effective learning in the current situation,
- the ability to recognize a class of learning problems, and to index the background knowledge for access to class knowledge.

The first three of the above six capabilities of an *L*-ML system relate to the use of knowledge to direct learning. Learning algorithms that have this characteristic are called *knowledge directed* (KD). KD algorithms need not be knowledge dependent, in the sense that pure explanation based learning (EBL) algorithms depend on having a complete domain theory.

The last three *L*-ML capabilities provide an observational discovery component for noting strong patterns in domain heuristics and conceptual regularities that come into play during learning. This type of introspective behavior has been termed self-watching. An *L*-ML system is thus a self-watching KD learning system that maintains working knowledge across multiple learning sessions by updating its background knowledge base.

Learning in *L*-ML systems can involve at least three different types of system metamorphosis in response to experience gained accomplishing prior learning tasks.

- Augmenting the concept language to be more expressive.

Some machine learning systems have extensible concept representation

1. This transformation is often called *constructive induction* [Michalski80].

2. Learning is directed by managing biases that are derived from meta-knowledge and goal structures found in the background knowledge, such as a Goal Dependency Network [Stepp86].

languages, languages in which frequently used or functionally integrated subconcepts can be denoted by a system-defined single symbol or predicate. Examples of such systems include CONFUCIUS [Cohen78], MARVIN [Sammut86], PLAND [Whitehall87], and SUBDUE [Holder88]. These systems store inductively derived concepts in a knowledge base. The system searches this knowledge base for prototype concepts when working on other problems. The concepts the system constructs in subsequent learning are related to the experiences it has had. Knowledge that augments the concept language is usually domain-specific but the domain characteristics are not encoded as preconditions for applying the knowledge (some other agent must ensure that the knowledge base is used only when relevant to a new problem).

Many similarity-difference based learning systems use fixed concept languages, fixed biases, and fixed background knowledge. Those that search for improved concept language and/or bias during learning normally do not add their improvements to a permanent knowledge store; the next run begins with the same initial language and/or bias. Examples of these systems include LEX [Mitchell83, Utgoff82], MIS [Shapiro81], STAGGER [Schlimmer87], INDUCE [Hoff83], and CLUSTER [Stepp86]. Many of these systems could be promoted to *L*-ML systems partly through the addition of a knowledge base manager to add bias knowledge to a permanent knowledge store.

- Chunking and transforming solutions to become more operational.

  Machine learning theory includes the distinction between learning at the knowledge level versus learning at the symbol level [Dietterich86]. There has been some debate about how to characterize the learning embodied in operationality transformations that change which and how many hypotheses a system can consider. In any event, such transformations do profoundly change the performance of the system on subsequent problem solving *and learning* tasks. They represent a second kind of metamorphosis that is

important for an *L*-ML system.

Operational knowledge can be applied to a problem using less search, thereby improving the system's performance against a fixed computation threshold. EBL systems (e.g., GENESIS [Mooney88], BAGGER [Shavlik88], PRODIGY [Minton87], and others) are noted for their focus on improving operationality. From the standpoint of *L*-ML systems, it is important to realize that the learning performance (as well as the problem solving performance) of EBL systems also improves when the generalized and operationalized schemas it produces are retained and available for subsequent learning.

Few similarity-difference based learning (SDBL) systems have the above characteristics, but there is nothing inherent in SDBL that prevents changing this. SDBL discovery systems (as opposed to discriminant concept generalization systems) are more likely to demonstrate operationality improvement because they profit from remindings of solutions to similar problems.

- Optimizing the learning algorithm by becoming more controlled.

  Learning algorithms are directed by internal (programmed) or external (user specified or knowledge based derived) control knowledge. Control knowledge includes heuristics, biases, feature selection rules, feature transformations[3], agenda management schemes, search control strategies, and hypothesis evaluation functions. Usually there is no obvious or direct relationship between control knowledge and the detailed composition of learned domain-specific concepts. This has often made the specification of control knowledge a black art.

  Learning systems that can discover relationships between problem domains

3. E.g., when to perform constructive induction and how to select the transformation rules.

and control knowledge will have solved one of the obstacles to the pervasive use of machine learning. To a degree, adjustable bias systems exhibit this type of *L*-ML behavior. Systems like STABB [Utgoff86], and VBMS [Rendell87] adapt the way they work by varying their biases. Although these systems contain the needed knowledge sensitive control features, at present only VBMS reports its control knowledge findings and associates them with characteristics of the problem domain, thus making the bias settings potentially available for subsequent reuse on similar problems. The typical bias adjusting algorithm rediscovers the proper choice of bias from scratch, for each application. The similarities between adapting control knowledge and automatic programming may eventually lead to an advantageous combination of automatic programming with machine learning.

One cornerstone of intelligence (and of *L*-ML systems) is the ability to discover. Crucial knowledge for improving learning system performance is found in the patterns and unsuspected relationships discovered in the course of learning. These patterns could be identified using conceptual clustering (e.g., CLUSTER [Stepp86] or COBWEB [Fisher87]) or a generalization based memory approach (e.g., UNIMEM [Lebowitz86]) and used during learning. Important characteristics to discover include patterns of empirical relationships in domain data, and patterns between domain data and the most effective general biases.

Some *incremental* learning systems can accept previous concepts as working hypotheses, and then improve them in light of new examples, using limited memory resources. The ability to use previous hypotheses gives each cycle of incremental learning a large performance boost when compared with repeated batch learning. A *L*-ML system should be able to do more: it should also take advantage of operationalized potential solutions, an extended concept language, and be able to utilize previously acquired concepts to compose new hypotheses, potentially combining several known concepts together.

2. AN ANALYSIS OF SYSTEMS WITH SOME *L*-ML BEHAVIOR

Although most learning systems do not have a self improving component, some notable systems do. For example *L*-ML behavior (with performance changes accumulating across problems from potentially different domains) is evidenced by many EBL systems. Such systems *discover* generalized schemas that are good shortcuts to problem solving, and also good shortcuts to learning. One or more learned generalized schemas may be combined to explain a new example. The EBL learner has learned by storing and indexing learned schemas. Chunking [Laird87] provides similar effects.

Among data-driven similarity-difference based learning systems there are few that demonstrate *L*-ML behavior. This is because many such systems use fixed or user supplied biases rather than knowledge-directed biases. Some SDBL systems that do use knowledge based adjustable biases and/or knowledge driven transformations (such as constructive induction [Michalski83b]) lack a way to store findings to help direct subsequent learning.

In this section, six contemporary learning algorithms are discussed with respect to their capabilities as *L*-ML systems.

2.1. EBL as represented by GENESIS

As discussed in [DeJong86], an important aspect of building a schema in EBL systems is the ability to use the new schema in the future. By using previously learned schemas the system is able to solve problems that would be beyond the processing capabilities of the system without those schemas. Another advantage is that they provide a mechanism for generalizing the structure of the example. The GENESIS system [Mooney88] (as a prototypical EBL system) improves its learning performance by using schemas it may have discovered previously.

GENESIS learns a schema to describe events in natural language stories. Consider GENESIS as it learns a schema for kidnapping given a story in which someone is held hostage by being threatened with a gun. The system possesses knowledge about *bargain*, *capture*, *threaten*, etc. in schemas the system has built

from previous examples. Using its deductive mechanisms the system is able to build a proof tree that explains the kidnapping event.

In this process some of the previously defined schemas are incorporated into the new explanation. Having to regenerate all schemas might well cause GENESIS to exceed its space/time limits, and thus do an inferior job of learning. Also, using previously defined schemas allows the system to increase the generality of the new schema. In the story, suppose John points a gun at Mary to force her into his car. The system recognizes this as an act of *capture* and uses the previously defined schema. If the capture schema were not in the system, then the new schema for kidnapping would only allow for a single method of abducting someone—with a gun. By using the previously learned generalized capture schema, the many ways a person may capture someone can be used to explain a new instance of kidnapping.

Shavlik [Shavlik88] has shown that EBL systems improve their performance by reusing learned schemas. His results indicate that the advantages of building new schemas from operationalized, previously acquired explanations outweigh the burden of searching a larger knowledge base. The results also indicate that the most general schemas built by his BAGGER system are the ones that can most decrease the learning time required. This is because fewer rules need to be found to cover all the cases.

2.2. Soar

Soar [Laird87] learns by chunking. The system stores its solutions to a search problem in long term memory, in the form of production rules. The chunking mechanism adds new rules to production memory after solving a previously unobserved problem successfully. This solution is generalized (in a way similar to EBL systems, but not as extensively) and may be called upon during the next cycle.

Soar uses its learned chunks to build new chunks. It performs *within-trial transfer*: a chunk found early in problem solving may be used as part of the

ultimate solution, as Soar continues to work on the same problem. Chunking is a form of learning by operationalization, like that done in EBL. The system profits from previous experiences by building new chunks from the solution found by its internal problem solver that uses previously acquired chunks to solve problems.

2.3. VBMS

The variable-bias management system (VBMS) [Rendell87] improves its learning performance by learning the proper bias to use for classes of problems. This approach to improving learning is significantly different from the methods mentioned above. EBL and Soar directly use the knowledge they have gained in creating new knowledge. They learn domain knowledge whereas VBMS learns meta-knowledge for modifying inductive bias. In VBMS, a region belief table (RBT) is used to indicate which bias point in the bias space is appropriate for the given problem. VBMS can improve its capabilities as it handles more problems by refining the RBT to make sharper distinctions between problems. The information in the RBT is not directly used in the solution of the problem, but rather controls the learning system's biases.

VBMS works by splitting the problem space into regions using the PLS1 algorithm. Problems are characterized by features and values that define global attributes of the class of problems being handled. The problem space is the set of all such problem points for the predefined features and their values. The problem belief table (PBT) contains all the biases explored for a specific problem and a measure of credibility for each bias. The system partitions the problem space into regions of points with similar PBT's. Every problem given to the system defines a point in the problem space and this point is contained within some PBT. Each PBT is defined within an RBT that indicates the type of biases that should be used for the problem. As the system sees more problems, the PBTs and RBTs are refined to improve the selection of bias for new problems which in turn allows the system to give better, faster results.

2.4. LAIR

The LAIR system [Watanabe87] incrementally learns conjunctive concept descriptions from examples by applying a domain theory for performing constructive induction [Michalski83a]. LAIR uses a hill climbing approach with limited incomplete memory that forces the system to forget all but the last seen positive example and the current working concept hypothesis.

LAIR's knowledge base consists of examples, concept descriptions, concept description constraints, and learnable domain knowledge. The knowledge base is built of frames and production rules. Rule frames in the knowledge base express first order implicative rules with literal consequents. On the other hand, concept descriptions determined by LAIR are lambda conjunctive formulas that are refined by the system as it learns to recognize correctly the class of positive examples.

By transforming a learned concept for a class into an implicative statement where the antecedent is the learned concept definition and the consequence is a predicate symbol identifying the class, the system can feed learned concepts into its rule base. For example if C is the learned concept description for the class "can-stack-on", then the rule C(x) => can-stack-on(x) could be captured in the rule base and used in subsequent learning. This potential capability of LAIR is mentioned by its author but is not illustrated with an example.

2.5. PLAND

The PLAND system [Whitehall87] discovers planning macro-operators (macrops) by observing sequences of executed actions. PLAND incorporates many of the abilities required of an *L*-ML system. The system uses previously learned structures to help discover new, more complex macro-operators. PLAND uses domain-specific background knowledge to guide the search for new macrops. And, the system is able to compose hypotheses based on relevant background knowledge by allowing separate contexts (or perspectives) to be considered at the same time.

A trace of observed actions describing the performance of a task is input to the PLAND system. From this trace, the SDBL system discovers macro-operators that consist of sequences, loops, and conditionals. If no background knowledge is applicable to the given trace, the system finds a regular grammar that describes the input, where the actions are treated as symbols of the language alphabet. With or without initial background knowledge of applicable macrops, PLAND is able to use newly discovered macrops in the course of further macrop generation. Such within-trial learning allows the system to build a hierarchical representation of the action trace and to discover macrops that would not be possible otherwise. As an example, let a trace of observed actions be denoted by the string *ABBBBDABBBBBDACCDACCCCDABBBD*. From this trace PLAND immediately discovers the loop constructs for B^* and C^*. These are then used to define the macrop for the whole input $(A\ (B^* + C^*)\ D)^*$, which would not be discoverable without the learned macrop components. Thus the performance of the system is improved by its own learning capabilities.

PLAND performs all the discovery processing within the confines of a *context*. A context is a data structure that contains the agendas for the context, the level of generalization used, and previously discovered macro-operators. An *agenda* defines a search operation for a specified type of macrop (loop or conditional) and specifies where within the input sequence the search should occur. Before any agenda is executed, background knowledge is used to check the applicability of the agenda. An agenda may be rejected if it operates on portions of the input sequence that the system has reason to believe are devoid of macrops or, for example, if it is looking for conditionals, and the system infers that conditionals are not appropriate within the observed task. This use of knowledge eliminates wasted search effort.

Knowledge is also used to select the context. When a context is selected, generalizations guided by background knowledge can be used to determine the attributes of actions that are considered relevant for action comparisons. For example, if actions X and Y each have some property A, then $XXXYYYYXXX$

could produce the macrop Z^* where Z denotes actions with the A property. By producing contexts with different levels of generalization, the system is able to work with proposed hypotheses. The generalizations of the context define the level of abstraction. Switchable contexts allow the system to work on more than one subproblem until a predominant solution emerges.

2.6. SUBDUE

SUBDUE is an L-ML system for discovering conceptual substructure in examples [Holder88]. The examples given to SUBDUE can be descriptions from a certain domain, descriptions of a knowledge base, descriptions of a goal structure, or any other group of structured knowledge representable in first-order calculus. With such input, SUBDUE can discover patterns, or substructure, in the knowledge and retain the substructures for use in subsequent learning tasks. The substructures discovered in the knowledge can be used to compress the knowledge base, form new features for constructive induction and concept language augmentation, and suggest rules for applying the knowledge to similar domains.

The SUBDUE system consists of a substructure discovery module, a substructure specialization module, and a substructure background knowledge module. The discovery module discovers substructures in the given input examples using a computationally constrained best-first search guided by four heuristics: cognitive savings, compactness, connectivity and coverage. These heuristics are motivated from results in gestalt psychology, data compression, and numerical and conceptual clustering. The specialization module specializes the best substructure found during the discovery process by adding additional structure to the substructure. The additional structure represents information about the context in which the substructure is applicable. Both the discovered and specialized substructures are stored in the background knowledge module. Within the background knowledge, substructures are stored hierarchically by defining the substructures in terms of previously defined, more primitive structures. During subsequent discovery tasks, the background knowledge module suggests

substructures from which to begin the discovery process.

As an example of SUBDUE, consider the input example shown in Figure 1a. After considering 29 alternative substructures, the best substructure discovered by SUBDUE is that shown in Figure 1b. Figure 1c shows the substructure after specialization. Both substructures are stored in the background knowledge. Now that SUBDUE has learned these new substructure concepts, they can be used to reduce the complexity of future examples containing the same substructures and improve SUBDUE's ability to discover more complex substructures. In addition, the newly discovered substructures augment the concept language with new, constructive features. The simplified example descriptions and constructive features can improve the speed and quality of results of other learning systems.

One of the machine learning areas providing great challenge is the area of learning concepts involving structured examples, especially the task of discovering structural concepts. The PLAND and SUBDUE systems show that some concepts cannot be learned until the system has learned simpler concepts from previous exercises. In this way SUBDUE and PLAND augment their own concept language and provide this augmented language to subsequent learning processes.

3. CONCLUSION

Knowledge-directed machine learning algorithms provide the advantages of SDBL and EBL approaches. Further power stemming from the application of

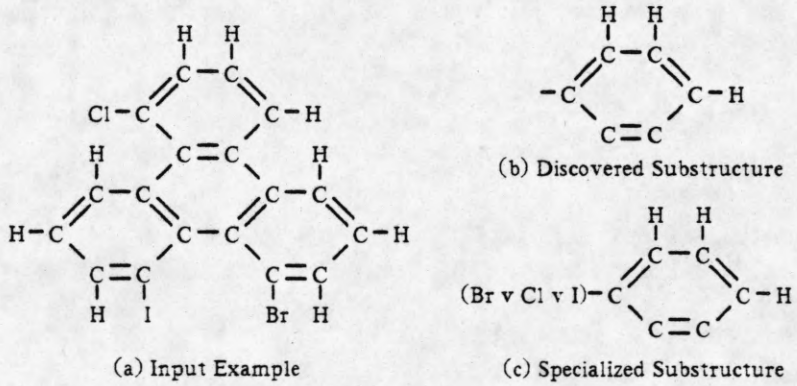

Figure 1. SUBDUE Example

machine learning techniques to the machine learning algorithms themselves could give *learning* machine learning systems important advantages over more limited current approaches.

It is instructive to note that current systems that have *L*-ML behavior fall mainly into two categories: EBL systems and SDBL discovery systems. In both kinds of systems, concepts are built by taking account of observations of the input phenomena. SDBL discovery systems create concepts from observed examples, augmenting the concept language in a way that is useful for interpreting new observations on subsequent learning. EBL systems use the observed training example(s) to improve the operationality of both themselves and a performance system. The VBMS approach is unique in its ability to optimize biases and heuristics based on discovered control knowledge. These system types exploit unequally different ones of the three main *L*-ML metamorphoses described in Sec. 1. Incorporating the metamorphoses in one system would create a powerful *L*-ML tool.

The chart in Figure 2 summarizes the major characteristics of the six learning systems that were presented. The table shows that a mechanism that chunks to improve operationality is provided (in some form) by all six algorithms. Also,

L-ML Prototype Systems						
feature	EBL	SOAR	VBMS	LAIR	PLAND	SUBDUE
B.K. transforms representation	no	no	no	yes	yes	yes
goal hierarchy in B.K.	yes	yes	no	no	yes	no
B.K. helps compose hypotheses	yes	yes	no	yes	yes	yes
discovers patterns	no	no	yes	no	yes	yes
updates B.K.	yes	yes	yes	yes	yes	yes
recognizes similar learning situations	no	no	yes	no	no	no
augments concept language to be more expressive	yes	yes	no	yes	yes	yes
chunks and transforms to be more operational	yes	yes	yes	yes	yes	yes
optimizes biases and heuristics to be more controlled	no	no	yes	no	no	no

Figure 2. *L*-ML Characteristics of Six Learning Systems.

each of these systems has some mechanism for updating a permanent knowledge base of domain and control concepts. The selection of representative learning systems for discussion here was based on evidence of a number of *L*-ML capabilities and to consider a wide range of approaches. With further development of such systems, there may soon be a time when systems possess all *L*-ML characteristics (and the focus will be on additional facets of intelligent learning behavior).

REFERENCES

[Cohen78]
B. L. Cohen, "A Theory of Structural Concept Formation and Pattern Recognition," Ph.D. Thesis, Department of Computer Science, University of New South Wales, Sydney, Australia, 1978.

[DeJong86]
G. F. DeJong and R. J. Mooney, "Explanation-Based Learning: An Alternative View," *Machine Learning 1*, 2 (April 1986), pp. 145-176. (Also appears as Technical Report UILU-ENG-86-2208, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)

[Dietterich86]
T. G. Dietterich, "Learning at the Knowledge Level," *Machine Learning 1*, 3 (1986), pp. 287-316.

[Fisher87]
D. H. Fisher, "Knowledge Acquisition Via Incremental Conceptual Clustering," Ph.D. Thesis, Department of Information and Computer Science, University of California, Irvine, Irvine, California, 1987. (Also appears as Technical Report 87-22)

[Hoff83]
W. A. Hoff, R. S. Michalski and R. E. Stepp, "INDUCE 3: A Program for Learning Structural Descriptions from Examples," Technical Report UIUCDCS-F-83-904, Department of Computer Science, University of Illinois, Urbana, IL, 1983.

[Holder88]
L. B. Holder, "Discovering Substructure in Examples," M.S. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, 1988.

[Laird87]
J. E. Laird, A. Newell and P. S. Rosenbloom, "SOAR: An Architecture for General Intelligence," *Artificial Intelligence 33*, 1 (1987), pp. 1-64.

[Lebowitz86]
M. Lebowitz, "Integrated Learning: Controlling Explanation," *Cognitive Science 10*, 2 (1986), pp. 219-240.

[Michalski80]
R. S. Michalski, "Pattern Recognition as Rule-Guided Inductive Inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence 2*, 4 (July 1980), pp. 349-361.

[Michalski83a]
R. S. Michalski and R. E. Stepp, "Learning from Observation: Conceptual Clustering," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell (ed.), Tioga Publishing Company, Palo Alto, CA, 1983, pp. 331-363.

[Michalski83b]
R. S. Michalski, "A Theory and Methodology of Inductive Learning," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, T. M. Mitchell (ed.), Tioga Publishing Company, Palo Alto, CA, 1983, pp. 83-134.

[Michalski86]
R. S. Michalski, "Understanding the Nature of Learning: Issues and Research Directions," in *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell (ed.), Morgan Kaufmann, Los Altos, CA, 1986, pp. 3-25.

[Minton87]
S. Minton, J. G. Carbonell, C. A. Knoblock and D. R. Kuokka, *The PRODIGY System: An Integrated Architecture for Planning and Analytical Learning*, Carnegie-Mellon University, Pittsburgh, PA, January 1987.

[Mitchell83]
T. M. Mitchell, P. E. Utgoff and R. Banerji, "Learning by Experimentation: Acquiring and Refining Problem-solving Heuristics," in *Machine Learning: An Artificial Intelligence*

Approach, R. S. Michalski, J. G. Carbonell, T. M. Mitchell (ed.), Tioga Publishing Company, Palo Alto, CA, 1983, pp. 163-190.

[Mooney88]
R. J. Mooney, "A General Explanation-Based Learning Mechanism and its Application to Narrative Understanding," Ph.D. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, January 1988. (Also appears as UILU-ENG-87-2269, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)

[Rendell87]
L. Rendell, R. Seshu and D. Tcheng, "Layered Concept-learning and Dynamically-variable Bias Management," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987, pp. 308-314.

[Sammut86]
C. Sammut and R. B. Banerji, "Learning Concepts by Asking Questions.," in *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell (ed.), Morgan Kaufmann, Los Altos, CA, 1986, pp. 167-192.

[Schlimmer87]
J. C. Schlimmer, "Incremental Adjustment of Representations for Learning," *Proceedings of the 1987 International Machine Learning Workshop*, Irvine, CA, June 1987, pp. 79-90.

[Shapiro81]
E. Y. Shapiro, "Inductive Inference of Theories from Facts," Technical Report 192, Yale University, New Haven, CT, Feb 1981.

[Shavlik88]
J. W. Shavlik, "Generalizing the Structure of Explanations in Explanation-Based Learning," Ph.D. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, January 1988. (Also appears as UILU-ENG-87-2276, AI Research Group, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.)

[Stepp86]
R. E. Stepp and R. S. Michalski, "Conceptual Clustering: Inventing Goal-Oriented Classifications of Structured Objects," in *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell (ed.), Morgan Kaufmann, Los Altos, CA, 1986, pp. 471-498.

[Utgoff82]
P. E. Utgoff and T. M. Mitchell, "Acquisition of Appropriate Bias for Inductive Concept Learning," *Proceedings of the National Conference on Artificial Intelligence*, Pittsburgh, PA, August 1982, pp. 414-417.

[Utgoff86]
P. E. Utgoff, "Shift of Bias for Inductive Concept Learning," in *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell (ed.), Morgan Kaufmann, Los Altos, CA, 1986, pp. 107-148.

[Watanabe87]
L. Watanabe and R. Elio, "Guiding Constructive Induction for Incremental Learning from Examples," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987, pp. 293-296.

[Whitehall87]
B. L. Whitehall, "Substructure Discovery in Executed Action Sequences," M.S. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, 1987. (Also appears as Technical Report UILU-ENG-87-2256)