From *Encyclopedia of Computer Science and Technology*, volume 11, pp. 24-51. Marcel Dekker, New York, NY, 1978.

MODELS OF LEARNING SYSTEMS

INTRODUCTION

Giving a machine the ability to learn, adapt, organize, or repair itself are among the oldest and most ambitious goals of computer science. In the early days of computing, these goals were central to the new discipline called cybernetics [2, 127]. Over the past two decades, progress toward these goals has come from a variety of fields—notably computer science, psychology, adaptive control theory, pattern recognition, and philosophy. Substantial progress has been made in developing techniques for machine learning in highly restricted environments. Computer programs have been written which can learn to play good checkers [93, 94], learn to filter out the strong heartbeat of a mother in order to pick out the weaker heartbeat of the fetus [125], and learn to predict the mass spectra of complex molecules [13]. Each of these programs, however, is tailored to its particular task, taking advantage of particular assumptions and characteristics associated with its domain. The search for efficient, powerful, and general methods for machine learning has come only a short way.

The terms adaptation, learning, concept-formation, induction, self-organization, and self-repair have all been used in the context of learning system (LS) research. The research has been conducted within many different scientific communities, however, and these terms have come to have a variety of meanings. It is therefore often difficult to recognize that problems which are described differently may in fact be identical. Learning system models as well are often tuned to the requirements of a particular discipline and are not suitable for application in related disciplines.

The term “learning system” is very broad, and often misleading. In the context of this article, a learning system is considered to be any system which uses informtion obtained during one interaction with its environment to improve its performance during future interactions. This rough characterization may include man/machine systems (see Ref. 64) in which humans take on active roles as required functional components. In some systems there is continuous interaction with the environment, with feedback and subsequent improvement. In other systems there is a sharp distinction between the interactions that constitute training and subsequent performance or predictions with no further training. Another way of differentiating between various learning systems is on the basis of what kinds of alterations they perform.

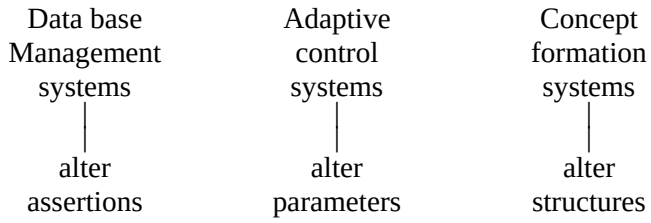

FIG. 1. Spectrum of learning systems.

Figure 1 shows several classes of systems that fit the above characterization, and lists the kinds of alterations which they perform. Data base systems are among the earliest kinds of systems which fit our definition. Such systems represent information about their environment by sets of alterable assertions. In the late 1950s and early 1960s, adaptive control techniques were first used to build programs which alter parameters in equations which model some aspect of the external world [93, 125]. The perceptrons of the early 1960s [71, 90] represent an attempt to use adaptive control techniques to train recognition networks by altering weighting parameters. More recently, concept formation (and other) systems have been written which build and alter structural representations as their model of the external world. In short, an important difference to be noted in LS's is in their internal representtations of the outer environment: some are mathematical models, some are linguistic assertions, and still others are structures encoding symbolic relations.

In this article, three distinct approaches to machine learning and adaptation are considered:

1. The adaptive control approach
2. The pattern recognition approach
3. The artificial intelligence approach

Progress in each of these areas is summarized in the first part of the article. In the next part a general model for learning systems is presented which allows characterization and comparison of individual algorithms and programs in all of these areas. Specific examples of learning systems are described in terms of the model.

ADAPTIVE SYSTEM APPROACH TO LEARNING

In the control literature, learning is generally assumed to be synonymous with adaptation. It is often viewed as estimation or successive approximation of the unknown parameters of a mathematical structure which is chosen by the LS designer to represent the system under study [18, 30]. Once this has been done, control techniques known to be suitable for the particular chosen structure can be applied. Thus the emphasis has been on parameter learning, and the achievement of stable, reliable performance [106]. Problems are commonly formulated in stochastic terms, and the use of statistical procedures to achieve optimal performance with respect to some performance criterion such as mean square error, is standard [130].

There are many overlapping and sometimes contradictory definitions of the terms related to adaptive systems. The following set, formulated by Glorioso [34], serves to illustrate the main features. An “adaptive system” is defined as a system which responds acceptably with respect to some performance criterion in the face of changes in the environment or its own internal structure. A “learning system” is an adaptive system that responds acceptably within some time interval following a change in its environment, and a “self-repairing system” is one that responds acceptably within some time interval following a change in its internal structure. Finally, a “self-organizing system” is an adaptive or learning system in which the initial state is unknown, random, or unimportant.

Adaptive control is an outgrowth of automatic control that has attracted significant research effort since the mid-1950s [3]. These investigations have been motivated by a desire for development of real-time control of incompletely known systems or “plants.” Limited plant specification is normally assumed to entail unknown, drifting parameters in a prescribed mathematical description. Various methods of adaptive control have been implemented for control of aerospace and industrial processes, as well as man-machine and socioeconomic systems.

Adaptive controllers have been coarsely divided into two large classes of active and passive adaptivity [113]. “Active adaptive” controllers are based on dual control theory [25]. In addition to the available real-time information, they utilize the knowledge that future observations will be made which will provide further possible performance evaluation and regulate their learning accordingly. “Passive adaptive” controllers utilize the available real-time measurements but ignore the availability of future observations. This limitation results in much simpler adaptive algorithms. Thus passive techniques have been much more extensively investigated.

Passive Controllers

Passive adaptive controllers can be subdivided into two classes: indirect and direct, denoting the primary focus of the adaptation mechanism either on plant parameter determination or control parameter determination, respectively.

Indirect adaptive control, originally suggested in Ref. 51, arbitrarily separates the control task into plant identification and control law calculation from the plant parameter estimates. This approach was designed to utilize the existing arsenal of control techniques requiring exact specification of the plant. Acceptance of this method has led to considerable interest in system identification [4]. Most parameter estimation schemes, however, are inherently open loop and suffer consistency and identifiability constraints when encompassed by feedback. This limitation can be circumvented by the injection of a perturbation input [95].

The alternative, which avoids the necessity of proper plant identification, is direct adaptive control, in which the available control parameters themselves are adjusted in order to improve the overall performance of the control system. Two broad techniques exist for establishment of convergent control parameter adaptation schemes: search methods and stability analysis. Search techniques generally suffer local convergence, whether based on gradient [40] or heuristic [30] methods. Alternatively, adaptive control algorithms arising from stability analysis can guarantee global asymptotic stability as a by-product. The widest application of stability theory

to adaptive control design has utilized Liapunov's second method [63]. The earliest application of Liapunov function synthesis for designing adaptive loops [98] utilized a model reference approach.

Model reference adaptive control techniques (see example in the Appendix) implement adjustment of reachable parameters in the overall controlled system so that its response to some reference signal exactly matches that of a predetermined model due to the same reference. Such a structural arrangement, in general, requires the ability to adjust each parameter independently in the overall controlled system. Assumption of this capability hampers the current sophisticated schemes of adapting feedforward and feedback parameters solely from plant input and output measurements [58, 74] by occasionally necessitating an unbounded control effort. Control effort boundedness is encouraged by abandoning exact output matching for input matching [48] which requires nonparametric, a posteriori determination of the optimal input.

No single adaptive control approach mentioned is without limitations in attempting to provide adequate control of a plant known only to be describable within a general structural class. The primary focus of adaptive control on parameter selection has led to provably convergent single level schemes. The ongoing merger of heuristic, layerable learning system concepts (as described below) with these convergent parameter adjustment algorithms of restricted applicability should improve the efficacy of adaptive control.

PATTERN RECOGNITION APPROACH TO LEARNING

Pattern recognition techniques are primarily employed at the interface of intelligent agents and the real world of physical measurements and processes. The interface attempts to provide some sensory capability to the agent, such as vision, touch, or some other nonhuman sensory modality. In this context, a "pattern" may be an image, a spoken word, a radar return from an aircraft, or whatever is appropriate to describe or classify a physical environment that is viewed through a particular set of sensors.

The problem of pattern recognition is often viewed as the development of a set of rules which can be used to assign observed patterns to particular known classes by examination of a set of patterns of known class membership. There are, however, a variety of related problems that can be discussed in the same framework. These include pattern classification, in which the classification rules are known, and the problem is simply assignment of patterns to classes, pattern formation, in which the classes themselves must be defined, and pattern description, in which the problem is to form descriptions (which are often symbolic in form) of the observed patterns, rather than assign them to classes.

The major concerns in pattern recognition are:

Convergence: the learning system should eventually settle on a stable set of rules, classes, or descriptions.

Optimality: the objective is minimization of some cost functional, such as the average risk associated with classification.

Computational complexity: the objective is minimization of the difficulty of using an algorithm, measured in terms of computation time, memory requirements, or programming complexity.

Pattern Recognition Subclasses

Pattern recognition is presently characterized by two major approaches. These are the statistical decision-theoretic or discriminant approach, which employs a classification model, and the linguistic (syntactic) or structural approach, which employs a description model. The first approach has been more extensively studied and a modestly large body of theory has been constructed, whereas the second approach is relatively new and many unsolved problems remain.

The decision-theoretic approach commonly involves the extraction of a set of characteristic (typically low-level) measurements, or "features," from a set of patterns. Each pattern is thus represented as a feature vector in a feature space, and the task of the pattern recognition device is to partition the feature space in such a way as to classify the individual patterns. Features, then, are usually chosen so that the "distance" (on some suitable metric) between patterns in the feature space is maximized [89]. This approach has been successful for applications such as communication of a known set of signal waveforms corrupted by some form of distortion, such as noise or multipath interference. However, it has been criticized because it is concerned only with statistical relationships between features, and tends to ignore other structural relationships which may characterize patterns [52].

The linguistic or structural approach has been developed in part to correct some of the difficulties seen in the decision-theoretic approach. With this paradigm, patterns are viewed as compositions of components, called subpatterns or pattern primitives, that are typically higher-level objects than the features of the decision-theoretic model. Patterns are often viewed as sentences in a language defined by a formal grammar (sometimes called a pattern grammar). Segmentation of patterns into primitives and formation of structural descriptions are thus the primary issues. This approach embodies an attempt to use other sources of information as aids to pattern recognition (e.g., in a speech-understanding system [21, 62, 87, 88, 91], syntax, semantics, and context act as powerful sources of information in addition to the recorded information).

In that both parametric and structural techniques are applied, pattern recognition effects a bridge between the adaptive systems and concept formation approaches to learning system design. We have recently begun to see a merger of the two approaches (see, for example Stockman [111]) which may result in more powerful systems. For a review of the current state-of-the-art, see Refs. 14, 53, 81, and 86.

The remainder of this section contains brief descriptions of major approaches to pattern recognition. Specific techniques are grouped according to their bias toward one of the two primary models: the classification model and the description model. Artificial intelligence research, discussed in the next section, has been a major factor involved in the movement away from complete adherence to the classification model and toward exploration of the description model.

Classification Model

In this model, patterns (feature vectors) are viewed as members of a class, and the aim is to assign observed patterns to classes. The classification may be

either statistical, wherein the patterns are thought to belong to one of a number of classes according to some set of probability density functions, or "fuzzy," wherein patterns are thought to have differing degrees of membership in a number of classes [131].

Variations

Classifiers may be categorized in a number of ways, depending on the type of classification rule and the sampling procedure they employ [45].

Classifiers may be categorized as parallel or sequential: parallel classifiers base their classifications upon the complete set of features extracted simultaneously during a single observation of a pattern, whereas sequential classifiers assign a pattern to a class on the basis of a sequence of observations. After each observation is made and integrated with past observations, a decision is made as to whether sufficient information has been gathered upon which to base a classification, or whether another observation must be made according to a test like the Wald Sequential Likelihood Ratio Test [119].

Classifiers may be further categorized as adaptive or nonadaptive. Adaptive classifiers (see example in the Appendix) are distinguished by the fact that their classification rules are themselves adjusted to improve performance as experience is gained with patterns drawn from the various classes of interest (a variety of procedures have been developed to adjust the rules—see, for example, Ref. 126). Nonadaptive classifiers, on the other hand, use a fixed set of classification rules and, in the language of this paper, are not considered to be learning systems.

Bayesian Classification

This type of classification is optimal in the probability of error sense. The strategy is minimization of the average risk of a classification, and complete knowledge of the a priori and conditional probability densities is assumed (where the a priori probability is the probability that a pattern is drawn from a particular class, regardless of its observed characteristics, and the conditional probability is the probability that a pattern with the observed characteristics could have been drawn from a particular class). The notion of "risk" arises because costs are assumed to be associated with different types of classification errors. When equal costs are assumed for all types of error, the result is the maximum a posteriori (MAP) classifier (where the a posteriori probability is the probability that a pattern has been drawn from a particular class, based on its observed characteristics).

Maximum Likelihood Classification

Likelihood is the conditional probability that the observed characteristics of a pattern indicate that the pattern should be assigned to a particular class. No knowledge of a priori probabilities is assumed, but the method does assume knowledge of the form of the density functions (e.g., Gaussian).

Nonparametric Classification

This type of classification does not guarantee the best possible performance but requires no knowledge of the underlying probability density functions that govern the generation of patterns. Techniques used in nonparametric classification include the K Nearest Neighbor Rule, which bypasses probabilities altogether, and assigns patterns to classes based on the proximity of their observed characteristics to those of neighboring patterns of known class membership, and the Fisher Linear

Discriminant, which is used to transform the feature space into another (decision) space (typically of lower dimensionality), in which parametric procedures can be employed [19].

Description Model

With this model, emphasis is placed on segmentation of the patterns into a set of meaningful primitives, and on generation of structural descriptions (generally symbolic in form) of the patterns. It is further assumed that a great deal of a priori knowledge of the pattern types that are of interest is available.

The approach is useful in applications like scene analysis [19, 65] where classification is clearly inappropriate. It also tends to be useful when the patterns themselves are complex [32], as it emphasizes hierarchical decomposition of patterns into their constituent components.

There is a variety of descriptive formalisms in which to express the structural descriptions. These include pattern grammars [31] and relational graphs [129]. Pattern grammars embody an attempt to carry over a large body of theory from the study of natural and programming languages. A variety of pattern grammars have been developed [52], both deterministic and stochastic in form. Relational graphs have been used in pattern recognition systems developed by the artificial intelligence community (see, for example, Winston [128]). Pattern primitives are taken as nodes in a directed graph whose edges indicate the relations between the primitives. Such graphs form a convenient representation for patterns with a high degree of hierarchical structure.

The text by Duda and Hart [19] serves as an excellent introduction to the methods used in the structural approach.

ARTIFICIAL INTELLIGENCE APPROACH TO LEARNING

In the 1950s and early 1960s there was considerable discussion of learning programs in the artificial intelligence (AI) literature (e.g., Refs. 23, 29, 70, 76, 80, 96, 102). It was hoped at the time that a general learning program could be written to accumulate and refine a large, detailed knowledge base about a domain [71]. That knowledge base, then, could be used by ever-improving high performance programs that reason in that domain. Samuel's programs that learn to play excellent checkers [93] were an early demonstration of success, but also demonstrated the amount of effort necessary to achieve success. On the reasons why learning tasks have been central in AI, Newell wrote [77]:

> Inductive tasks have always been a prominent part of the artificial intelligence landscape. The reasons for this seem to be twofold. For one, we have inherited a classic distinction between deduction and induction, so that the search for intelligent action should clearly look to induction. Second, American psychology has largely identified the central problem of conceptual behavior with the acquisition or formation of concepts—which in practice has turned out to mean the induction of concepts from a set of presented exemplars.
>
> This tendency, shaped strongly by Bruner, Goodnow, and Austin's Study of Thinking [11], derives fundamentally from the emphasis on learning that has characterized American psychology since the rise of behaviorism.

The motivation for writing these programs is diverse. Some are written as testable psychological models of how human subjects perform a learning task (e.g., Refs. 23, 46, 47, and 104); others are written to demonstrate the feasibility of a method (e.g., Ref. 109), and still others are written with the express purpose of aiding human problem solvers codify and explain data (e.g., Ref. 13). Insofar as all the programs mentioned below perform well at their stated tasks, they all illustrate the emerging power of heuristic programming methods for improving the problem-solving power of computer programs.

All the AI learning programs written to date have strong limitations on their generality. Some are applicable to just one kind of problem, others work with several types of problems within a larger class defined by the representation of objects and relations in the domain.

Early AI research was closely tied to pattern recognition and the adaptive systems approach (see, for example, Refs. 97, 116, and 117). For example, much work has been performed on learning automata [79] (see also Ref. 75) and neural networks that grow in response to stimuli [71]. All of these efforts have aimed at defining simple machines that learn to respond to their environments [27]. Newell [77] traces one line of growth from stimulus-response learning in psychology to (1) pattern recognition and self-organizing systems, as well as to (2) concept formation, induction, and other AI work. The two fields diverged in the 1960s, and are now quite distinct. Whereas pattern recognition and control research emphasizes adjustment of parameters, AI research emphasizes construction of symbolic structures, based on conceptual relations. For example, Feigenbaum's EPAM program [23] used a discrimination net (i.e., a tree of tests and branches) to store the relations required to recall nonsense syllables in a rote learning experiment (see Refs. 26, 112, and 129 for further examples).

In AI it is commonly believed that a learning system should have sufficient internal structure to develop a "strong theory" of its environment [24, 64, 72]. Much emphasis has therefore been placed on building "knowledge-based" or "expert" systems that not only have the capacity for high performance but can also explain their performance in symbolic terms [17].

Winston [129] describes various levels of sophistication in learning systems: learning by being programmed, learning by being told, learning from a series of examples, and finally learning by discovery. We see in this categorization a gradual shift in responsibility from the designer/teacher to the learning system/student. At the highest level, the system is able to find its own examples and carry on autonomously; at the lowest level the system is learning only in the sense that a programmer is explicitly programming it to do something.

The formalism of inductive inference has captured much attention also (e.g., Refs. 38, 44, 66, 67, 84, and 108). The purpose of much of the work on abstract formalisms is to find general principles of induction that can be mechanized. This was also a goal of Bacon and Leibniz centuries ago.

Considerable work is still expended on the Leibnizian dream of an abstract formalism for scientific inference. Some of this work is done specifically with computer programs in mind. Much of it, however, is done in abstraction. Programs based on these formalisms form hypotheses from data without any special knowledge of the domain from which the data were collected. The drawback of very general methods is that, while they may produce some interesting empirical generalizations, they are likely to produce many generalizations that experts in the domain would

regard as trivial or meaningless. In short, they lack a working model of the domain to guide judgments of plausibility.

Some recent programs explicitly recognize the need for problem-specific constraints. The Meta-DENDRAL program [13] discovers general rules about the behavior of chemical compounds in an analytic instrument known as a mass spectrometer. The data are noisy, they do not come already classified, the space of possible explanations is very large, and there is no single correct answer. Nevertheless, the program finds regularities in these data and formulates general rules to explain them.

The AQVAL program [60] accepts a set of descriptions of objects, and produces rules that can correctly classify these objects and others like them. For example, for descriptions of eastbound and westbound railroad cars containing circles, triangles, rectangles, etc., the program is able to find the shapes and relations among shapes that discriminate the two trains.

Still another program, named Thoth-pb [118], is able to learn rules for (1) extending letter sequences, (2) recognizing geometric analogies, (3) relating "before and after" situations, and (4) relating sequences of situations. It uses background knowledge about the domain to help it recognize important relations among features of the objects themselves.

Game Playing

Much of the work with learning systems in AI research has been done in the context of games. Improvement of the game-playing program is the ostensive goal, but the learning task itself is often the reason for the work (see, for example, Ref. 78). The nature of the learned information ranges from parameters governing the evaluation of moves (and ultimately their selection) to symbolic rules expressing how to play well in different situations.

Samuel's work is best known in this field [93, 94]. In the context of a checker-playing program, he has explored rote learning, parameter tuning, and building "signature tables" which are clusters of dependent features with weights that can be used to evaluate moves (cf. Refs. 37, 124). Waterman [121] compared the performance of a poker-playing program after learning with a human teacher and automated learning. The program represented its heuristics of good play in a table of conditional rules, or productions, which the learning system altered in light of mistakes. Waterman has generalized many of these ideas to other tasks [122]. Findler [28] has also studied the game of poker. Pitrat's work on learning patterns in chess [83] applies many heuristic search ideas to learning useful combinations from examples of given games. Programs have also been written to learn dominoes [107], go-moku [20], and rules of tic-tac-toe [85]. Banerji [7] has studied learning processes for several classes of games and puzzles from a more formal point of view. Koffman [54] has also related game playing to pattern recognition.

Concept Formation

In concept formation tasks, a computer program (or human subject) is presented with objects, or descriptions of objects, which exhibit a common concept. The program (or subject) is expected to generalize from these instances well

enough to classify new objects accurately. Negative instances—i.e., objects which fail to exhibit the concept—are sometimes presented to the program (and identified as negative instances) in addition to the exemplars of the concept. When training includes negative instances, learning is faster and more accurate. Concept formation has long interested psychologists as a learning task. As with other learning tasks, computer programs have been written to simulate the performance of human subjects—and thus test a psychological model [104]. Or they have been written to learn mechanisms other than those humans use—and thus demonstrate some modicum of intelligence on the part of computers.

Two frequently cited AI concept formation programs are those written by Evans [22] and Winston [129]. Evans' program finds analogies among geometric figures to solve standard intelligence test problems of the form A is to B as C is to _ (pick one of Dl, D2, D3, D4, D5). The concept here is a transformation or rule which maps figure A into B and also maps figure C into one of the answer choices.

For Winston's program the task is to produce a correct description of a concept exhibited in a set of line drawings of block figures. An important feature is the introduction of "near misses," i.e., figures that fail to exhibit the concept because they differ with respect to a small number of essential properties. The program learns the correct description of an arch, for example, from descriptions of two posts and a lintel (exemplar), and of near misses such as T's and posts with a fallen lintel.

Another recent program learns concepts, such as hit and out, for the game of baseball [109] from a set of descriptions of events over the span of a game. Other concept formation programs are described in Refs. 13, 41-43, 45, 50, 59, 60, 73, 92, 104, and 132.

Grammatical Inference and Sequence Extrapolation

Grammatical inference and sequence extrapolation have often been taken as prototype induction problems. The task is to find a rule (or set of rules) that can serve as the generating principle of a training set of symbol strings. For example, the training instances may be the following allowable "sentences" in a hypothetical language: A, AB, ABB, ABBB. An uninteresting set of rules is just the training instances themselves. Without some generalization from the training instances, prediction of new sentences is impossible. The following two rules, then, will serve to define the grammar of which these strings are correct sentences:

(R1) A ["A" alone is a sentence]

(R2) $A \rightarrow AB$ ["A" can be replaced by "AB"]

The sequence extrapolation task is similar: given a sequence of symbols (usually but not always numerals) such as 1, 3, 5, 7, 9, find a rule which allows correct prediction of the next member of the ordered sequence. In this case, the generating principle is

(R3) n-th member = $2n - 1$

Both of these problems exhibit many characteristics of scientific hypothesis formation. Regularities in the data must be found and characterized, different generating principles must be proposed and tested, and alternative hypotheses must be ranked, for example by simplicity. Most programs [10, 82] assume the

initial data are free of errors. Many of these programs explicitly search a space of hypotheses (e.g., Cook's grammatical inference program [16]), but most recent work on grammatical inference emphasizes more formal methods [10, 33, 35].

Inferring natural language [100] and simple computer programs from examples are other induction tasks that have been studied [39, 99, 122, 123] using AI techniques. The training instances are often input-output pairs, and the task of the induction system is to find the rule (procedure) that will produce the specified output symbols for each associated input. While the tasks are similar to concept formation and grammatical inference, the languages are so much richer that progress is slow.

A MODEL OF LEARNING SYSTEMS

This section is concerned with a simple functional model that is useful for characterizing, comparing, and designing learning systems (LS's). Many of the functional components of an LS are essential to intelligent problem-solving systems in general, as noted by Simon and Lea [105]; that is, learning (induction, concept formation, etc.) is problem solving of one kind, which means that AI problem-solving methods and representations can be expected to apply to this task as well as to others.

Effects of the Environment

The environment from which training instances are drawn, and in which an LS operates, may have a profound effect upon the LS design. LS environments can be divided into two major categories: those that provide the correct response for each training instance (supervised learning) and those that do not (unsupervised learning). Supervised learning systems operate within a stimulus-response environment in which the desired LS output is supplied with each training instance. Examples include Samuel's "book move" checkers program [93, 94] and grammatical inference programs [45].

Unsupervised LS's operate within an environment of instances for which the correct response is not directly available. The version of Samuel's program which learns by playing checkers against an opponent falls into this category [93] since moves are not classified by opponents as, say, excellent, good, poor, or terrible. Learning systems operating within this type of environment must themselves infer the correct response to each training instance by observation of system performance for a series of instances. As a result, assignment of credit or blame for overall performance to individual responses is generally a problem for these systems [70]. Tsypkin [115] has pointed out that unsupervised learning is somewhat of an illusion in the sense that a teacher/designer defines the standards which determine the quality of operation of the LS at the outset, whether or not he is present during the actual operation of the system.

Environments can be further categorized as "noise-free" or "noisy." Noise-free environments, such as that of Winston's structural description learning program [129], provide instances paired with correct responses which the system assumes to be perfectly reliable. Most AI systems assume noise-free environments. (One exception is described in Ref. 13.) Noisy environments, on the other hand, do not

provide such perfect information, as is usually the case when empirical data are involved. Pattern recognition and control systems frequently operate within noisy environments [8, 18, 19].

The Model—Overview

The proposed LS model is shown in Fig. 2. The "performance element" is responsible for generating an output in response to each new stimulus. The "instance selector" selects suitable training instances from the environment to present to the performance element. The "critic" analyzes the output of the performance element in terms of some standard of performance. The "learning element" makes specific changes to the system in response to the analysis of the critic. Communication among the functional components is shown via a "blackboard" to ensure that each functional component has access to all required system information, such as the emerging knowledge base. Finally, the LS operates within the constraints of a "world model" which contains the general assumptions and methods that define the domain of activity of the system.

The components of the model are conceptual entities which specify functions that must be performed to effect learning. They simplify the characterization of existing systems, and will assist designers in the construction of new systems. Although the functional decomposition suggested by the model is not necessarily reflected in the physical decomposition of many existing systems, the model is useful for comparing systems and may aid in future learning system designs.

The following sections present detailed discussions of the LS model components shown in Fig. 2. In addition, the Appendix contains detailed characterizations of representative Al, pattern recognition, and control systems in terms of the model. The reader may find it helpful to refer occasionally to the Appendix while reading the following sections.

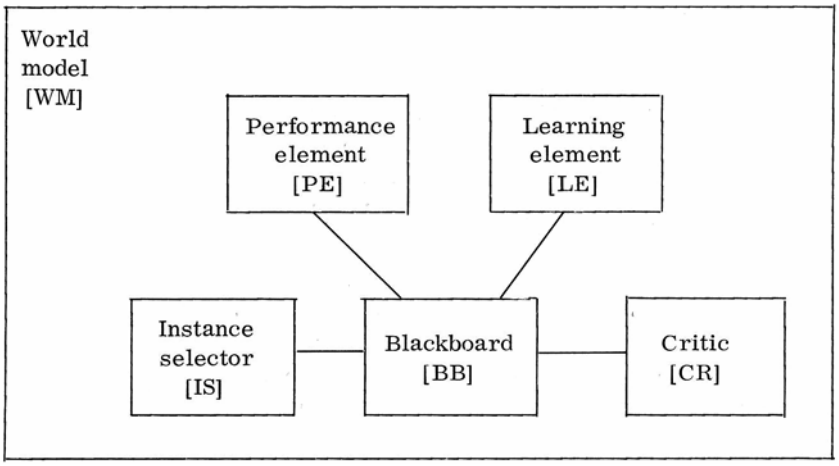

FIG. 2. The components of a learning system.

Performance Element

The performance element uses the learned information to perform the stated task. It has been included in the LS model because of the intimate relationship between what information is to be learned and how this learned information is to be used.

Performance elements are usually tailored more to the requirements of the task domain than to the architecture of the LS. In general, the performance element can be run in a stand-alone mode without learning, independent of the rest of the LS. In any LS, however, the ability to improve performance presupposes a method of communicating learned information to the performance element. Since its architecture must allow learned information to affect its decisions, additional constraints are placed on the performance element within an LS. The performance element should be constructed so that information about its internal machinations is readily available to the other system components. This information can be used to make possible detailed criticism of performance, and intelligent selection of further instances to be examined by the system.

The performance elements of existing systems also vary in the ways they may be altered by learning. For example, systems whose operation is determined by a set of production rules [121, 122] have the potential to exhibit richer variations than systems whose operations are keyed only to the adjustment of parameter values [57, 69].

Instance Selector

The instance selector selects training instances from the environment that are to be used by the LS. It is a functional component not clearly isolated in earlier adaptive system models.

In existing LS's, methods for instance selection vary mainly along the dimensions of responsibility and sophistication. The responsibility for instance selection varies between the extremes of completely external ("passive") selection and completely internal ("active") selection. In psychological experiments on concept formation, instance selection is closely controlled by the experimenter and the subject is completely passive in this respect. Instance selection in Samuel's book move checkers program [93] is externally controlled, whereas Popplestone's program [85], which learns the features that characterize a winning position in tic-tac-toe, generates its own training instances. It forms alternate hypotheses and then generates instances to choose among them (relying upon an external critic to evaluate these instances). (See also Ref. 105.) In the adaptive systems literature, Tse and Bar-Shalom [114] use a form of active instance selection known as "dual-control." They adjust the input to a system in such a way as to simultaneously control its output and obtain information about its internal structure.

The degree of sophistication used for LS instance selection is also an important consideration. In order to qualify as sophisticated, an instance selector must be sensitive to the current abilities and deficiencies of the performance element and must construct or select instances which are designed to improve performance. Winston [129] has shown the advantages to be accrued through presenting carefully constructed examples and "near-misses" of the concepts to be acquired by an LS. In general, careful instance selection can improve the reliability and efficiency of

an LS. It is important to note, however, that this may not always be permitted by the environment in which the LS operates, as is generally the case for adaptive control systems [18].

Critic

The critic analyzes the current abilities of the performance element. It may play three roles: evaluation, localization, and recommendation. The critic always operates as an evaluator in that it embodies a standard by which to assess the behavior of the performance element. This is the role that has been emphasized in earlier adaptive system models [30, 34, 106]. Feedback from a critic, at least as evaluator, is essential for learning.

The critic may also localize errors and localize the reasons for poor performance. This type of behavior is essential for resolution of the credit assignment problem described by Minsky [70]. In its diagnostic role, the critic is exemplified by the bug classifier and summarizer in Sussman's HACKER [112].

Finally, the critic may recommend repairs by making specific recommendations for improvement or suggestions about future instances. In Waterman's poker player [121], the critic in his role suggests the bet that should have been made by the performance element for a particular training instance. The critic not only recognizes poor play and isolates the production rules responsible for it, but suggests specific corrections so the program will not play as poorly in similar future situations.

The dividing line between critic and learning element is not sharp, and it is certainly possible to view therapy as a function of either the learning element or the critic. However, in mapping existing LS's into this model, we have adopted the convention that the critic's recommendations to the learning element are at an abstract level removed from the implementation considerations such as data representation. This clearly separates the two different functions of deciding what kind of change is needed and deciding how to implement that change.

In some LS's the functions of the critic have been left to humans. For example, MYCIN/TEIRESIAS [17] uses a human critic for evaluation, localization, and recommendation. The performance program applies rules (to cases selected by humans) and a human supplies criticism of results, localization of blame, and suggestions for altering the rule base. Because the computer program assists the user in these tasks, the learning can be said to be semiautomated.

Learning Element

The learning element is an interface between the critic and the performance element, responsible for translating the abstract recommendations of the critic into specific changes in the rules or parameters used by the performance element.

Representations for learned information exhibit great variety. They include, for example, production rules [121], parameterized polynomials [93], executable procedures [112], signature tables [94], stored facts [23], and graphs or networks

[129]. The method of incorporating new learned information is dependent upon the representation, and even among systems which use similar representations, competing methods are found (contrast, for example, Refs. 13 and 121).

The extent to which the learned information is altered in response to each training instance is an important LS design consideration. In some systems the learning element incorporates exactly the information supplied by the critic [129]. Were the same training instance to occur later, the response of the performance element would be exactly as the critic advised for the first occurrence. This type of learning is well suited to environments which provide perfect data and to systems with reliable critics. Under these conditions the LS will converge rapidly to the desired behavior. If such a system were provided with an incorrect classification by the environment or less than reliable advice by the critic, however, it might commit itself to incorrect assumptions from which it could not recover. Systems which make less drastic changes to the learned knowledge on the basis of a single training instance are less vulnerable to imperfect information, but consequently require more training instances to converge to the desired behavior. Many statistical LS's fall into this category [79]. Other systems consider several training instances at a time in order to minimize the effect of occasional noisy instances [13].

Blackboard

The blackboard of this model is a global data base which also functions as a system communications mechanism. It is similar to the concept introduced in the HEARSAY system [62]. The blackboard holds two types of information: the information usually associated with the "knowledge base" in AI programs and the temporary information used by the LS components. The knowledge base often contains the set of rules, parameter values, symbolic structures, and so on, currently being used by the performance element. Such information can be used as an aid to sophisticated instance selection if it is readily available. The temporary, system-oriented information includes, for example, the intermediate decisions made by the performance element in selecting a particular response. Detailed criticism by the critic is dependent upon the availability of this information.

In many existing systems this information is not so clearly separated or defined. The communication links between functional components, especially, are often programmed directly. Because the same information is required by many of the individual functional components of any LS, however, a blackboard is a more transparent communications mechanism.

World Model

Whereas the blackboard contains information that can be altered by the LS components, the world model contains the fixed conceptual framework within which the system operates [15]. The contents of the world model include definitions of objects and relations in the task domain, the syntax and semantics of the information to be learned, and the methods to be used by the LS. Among task domain definitions are, for example, the rules of a game and the representation of inputs and outputs for the performance element. This part of the world model simply defines the task of the performance element and the standard of performance (the evaluation

function) to be applied by the critic. Domain specific heuristics are also commonly added to the world model of AI systems to guide inferences made by the LS (e.g., heuristics about the world of blocks in Winston's program [129]). Definitions of the syntax and semantics of information to be learned define the mode of communication between the learning and performance elements.

The assumptions and constraints from which the world model is composed are of critical importance in the design and characterization of LS's. Although many of these assumptions are often hidden in the various functional components, the LS designer and user must both be aware of each of them. We believe that, where possible, world model constraints should be made explicit in order to allow for their modification during the design process.

Multilayer Learning Systems

Although the world model cannot be altered by the LS that uses it, the designer can alter its contents in order to improve LS performance. He often changes parameters and procedures of the basic LS after observing and criticizing its behavior for some carefully chosen training set. These alterations result in a new version of the LS, which is then tested on some training set, and so on. The designer views the whole LS as a system whose performance needs improvement, and he selects instances, criticizes performance, and makes changes accordingly. In other words, the designer's activities can be modeled by a system whose components are just those of Fig. 2. This leads us to the concept of layered LS's, each higher layer able to change the world model (vocabulary, assumptions, etc.) of the next lower layer on the basis of criticizing its performance on a chosen set of instances. Thus adjustments can be made to the world model of some learning system LS1 by another learning system, LS2, which has its own functional components (critic, world model, etc.), as shown in Fig. 3. In turn, it is conceivable that a third system, LS3, could adjust the world model of LS2, and so on. The designer constitutes the final critic of course, operating above the "top-level" LS. Each lower layer constitutes the performance element of the next higher layer, and interlayer communication is effected through the blackboards of the various layers. The use of a blackboard in the single layer LS model was partly motivated by its attractiveness in the multilayer context.

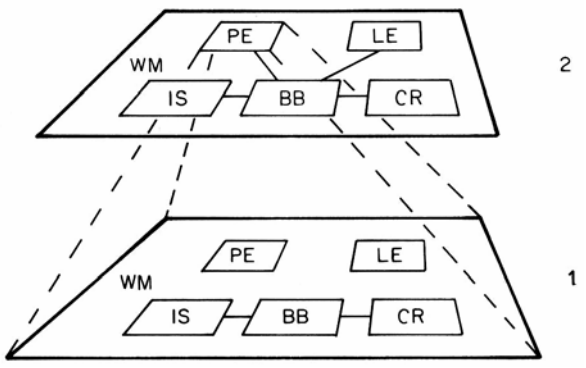

FIG. 3. Layering of learning systems. (Components are labeled as in Fig. 2.)

This multilayer architecture involves bidirectional information passing; that is, the effects of adjustments made in a layer may propagate both to lower and higher level layers. It is a hierarchical architecture, in the general sense [103], and includes as a specific case the bottom-to-top hierarchical architecture used, for example, by Soloway [110].

One existing LS which may be viewed as a layered system is the version of Samuel's program [94] which learns a polynomial evaluation function for selecting checkers moves (see the Appendix for details). The lower layer (LS1) in this system adjusts the coefficients of a given set of game board features in order to improve performance of the move selection program. The second layer system (LS2) adjusts the set of board features used in the evaluation function in order to improve the performance of LS1. Since LS1 is contained in LS2 as the performance element, all the assumptions necessary for its operation also belong to the LS2 world model. In addition, the LS2 world model contains assumptions about the set of allowable game board features and the standard for evaluating LS1 performance.

A single layer LS, then, can never move outside its world model to make radical revisions to its way of viewing the task to achieve a "paradigm shift," as discussed by Kuhn [56]. However, a shift in the conceptual framework of LS1 could be made by a properly programmed LS2 [12]. We believe that a layered approach such as that described above provides a useful system organization for learning at various levels of abstraction in complex domains. Although there are examples of this kind of layering in the literature [93, 110, 117], no one has carried it as far as the model suggests. In fact, single layer learning systems are just now becoming well enough understood to consider developing more sophisticated systems.

Implications of the Model

The LS model described here provides a common language for characterization and comparison of different types of learning systems which operate in a variety of task domains. The model is a useful conceptual guide for LS design because it isolates the essential functional components and the information that must be available to these components.

A number of desirable features for future learning system designs are brought out by this model. First, the design should be modular, with individual modules corresponding to the functional components shown in the model. The knowledge used by the system should be made explicit and collected, as much as efficiency considerations permit, in a world model component. Especially the parts of the LS that are to be adjustable must be explicitly exposed. Intelligent criticism is important, as is active instance selection, although neither has been isolated as a separate object of study. Finally, a multilayer architecture for learning at different levels of abstraction is suggested by the model as a way of introducing still more intelligence into the whole learning system.

APPENDIX: CHARACTERIZATION OF EXISTING SYSTEMS

In this appendix several existing LS's are characterized using the framework provided by the model described in the section entitled A Model of Learning Systems. The systems selected are representative of several approaches to machine

learning. Because the blackboard contains information in a state of flux, its contents are not specified explicitly for the systems characterized below.

Model Reference Adaptive Control [57]

Purpose. Construct a "controller" which preprocesses inputs to an existing system (called the "plant"). The behavior of the combined controller-plant system is to mimic the behavior of a third system (called the "reference model") on the training data.

Environment. The plant to be controlled and the set of possible inputs (including disturbances).

Performance element. The controller is a system whose output is used as input to the plant. Its behavior is a function of the input signal, past I/O behavior of the plant, and a set of adjustable parameters.

Instance selector. Accepts data sequence (as input to the controller) from the environment.

Critic. Evaluation: Applies a measure of performance which is some function of the arithmetic difference between the plant and reference model outputs. In some cases the reference model is mathematically defined, and can therefore be considered part of the critic. In other cases the reference model is an actual system, and is considered part of the environment.

Learning element. Modifies the parameters of the performance element (controller), depending on the performance measure supplied by the critic.

World model. Control theory assumptions (time invariance, linearity, etc.) and techniques, and the standard of performance embodied in the critic.

Adaptive Pattern Classifier [55]

Purpose. Learn the parameters of a classifier that can classify a set of patterns in such a way as to minimize a specified cost functional.

Environment. Patterns drawn from a prespecified set of classes. Each pattern is represented as a feature vector.

Performance element. A linear pattern classifier which forms the inner product of a pattern feature vector (which constitutes the input) and a weight vector (where the weights constitute the adjustable parameters of the classifier). Based on the resultant scalar value, the classifier assigns the pattern to a class.

Instance selector. Accepts instances from a human trainer. The classifier uses a set of patterns of known class membership to tune the weights. Thereafter, the weights are held constant.

Critic. Evaluation: Computes the difference between the output value of the classifier and the known acceptable output (the learning in this example is supervised).

Learning element. Modifies the weights used by the classifier according to the LMS algorithm [126], based on the information received from the critic. This algorithm attempts to adjust the set of weights so as to minimize the mean-square error between the output of the classifier and the desired output.

World model. Pattern recognition assumptions concerning the suitability of representing the patterns as feature vectors, the suitability of a statistical formu-

lation of the classification problem, the suitability of a linear pattern classifier, the suitability of the selected performance measure, and the specific adaptation algorithm.

Checker Player [93, 94]

Purpose. Learn to play a good game of checkers [here we discuss only the version of the program which learns a linear polynomial evaluation function by examination of moves suggested by experts ("book moves")].
Environment. Set of all legal game boards.

LS1 (lower layer)

Purpose. Learn a good set of coefficients for combining board features in a linear polynomial evaluation function.
Performance element. Uses the learned evaluation function to rank plausible moves for a given board position.
Instance selector. Reads instances from a list of predefined game-board/recommended-move pairs.
Critic. Evaluation: Examines the ranking given to the book move by the performance element. Diagnosis: Suggests that the book move should be ranked above all other moves.
Learning element. Adjusts weights of linear polynomial to make move selection correspond to the critic's recommendation.
World model. Syntax of game board, form and features of linear polynomial evaluation function, method for adjusting evaluation function, and rules of checkers.

LS2

Purpose. Improve the performance of LS1 by selection of a good set of board features.
Performance element. LS1.
Instance selector. The entire set of possible training instances is simply passed to LS1 (via the blackboard).
Critic. Evaluation: Analyzes the learning ability of LS1 (i.e., the LS2 performance element) with the current set of evaluation function features. Diagnosis: Singles out features which are not useful. Therapy: Selects new features from a predefined list to replace useless features.
Learning element. Redefines the current set of features as recommended by the critic.
World model. The LS1 world model, plus the set of features which may be considered, and the performance standard employed by the LS2 critic.

Poker Player [121]

Purpose. Learn a good strategy for making bets in draw poker.
Environment. Set of all legal poker game states.
Performance element. Applies the learned production rules to generate actions in a poker game, e.g., bets.

Instance selector. Selects each game state derived by play against an opponent as a training instance.

Critic. Two versions of the program use two different critics. In both cases the critic performs the following functions. Evaluation: Decides whether the poker bet made by the performance element was acceptable. Diagnosis: Gives important state variables for deciding the correct bet. Therapy: Provides the bet which the performance element should have made. In "explicit" learning the critic is an expert poker player, either human or programmed. In "implicit" learning the evaluation and therapy are deduced from the next action of the opponent and a set of predefined axioms, while diagnosis is read from a predefined "decision matrix."

Learning element. Modifies and adds production rules to the system. Mistakes are corrected by adding a new rule in front of the rule responsible for the incorrect response.

World model. Rules of poker, features used to describe the game state, the language of production rules, heuristics for updating the rule base, the model of an opponent.

Meta-DENDRAL [13]

Purpose. Learn to predict data points in the mass spectra of molecules.

Environment. Set of all known molecule/data-point pairs.

Performance element. Predicts peaks (data points) in mass spectra of molecules using learned production rules. Employs a model of mass spectrometry for translating between mass-spectral processes (predicted by the rules) and data points in the spectrum.

Instance selector. Accepts a set of known molecule/spectrum pairs from the user.

Critic. Evaluation: Determines the suitability of the set of predictions generated by a rule. Diagnosis: States whether the rule is acceptable, too specific, or too general. Therapy: Recommends adding or deleting features to the left-hand sides of rules.

Learning element. Conducts a heuristic search through the space of plausible rules using a predefined rule generator. At each step in the search the potential rule's performance is reviewed by the critic.

World model. Representation of molecules as graphs, production rule model of mass spectrometry, vocabulary of rules used to represent learned information; heuristics used by the critic in directing the rule search.

Learning Structural Descriptions from Examples [128, 129]

Purpose. Learn to identify blocks world structures (such as arches and towers).

Environment. Set of possible line drawing/structure-classification pairs.

Performance element. Decides class of structures to which the input structure belongs. Uses a model of the structure class supplied by the learning element.

Instance selector. Accepts training instances supplied individually by the user.

Critic. Evaluation: Compares the classification made by the performance element against the correct classification as supplied with each training instance.

Diagnosis: Generates a comparison description pointing out differences between the model and the structure description.

Learning element. Constructs a model of the class of structures under consideration. Examines the comparison description supplied by the critic, and modifies the model to strengthen or weaken the correspondence between the model and the training instance.

World model. Representation of scenes as line drawings, method of translating line drawings to graphical descriptions, grammar for drawings to graphical descriptions, grammar for representing the learned information, domain-specific heuristics for resolving among possible changes to each structure class model.

ACKNOWLEDGMENTS

Supported in part by the National Institutes of Health, the Advanced Research Projects Agency, and the Department of National Defence of Canada.

REFERENCES

1. Arkadev, A. C., and E. M. Braverman, Learning in Pattern Classification Machines, Nauka, Moscow, 1971.
2. Ashby, W. IL, An Introduction to Cybernetics, Wiley, New York, 1963 (first published 1956).
3. Asher, R. B., D. Andrisani, II, and P. Dorato, Bibliography on adaptive control systems, Proc. IEEE 64(8), 1226-1240 (1976).
4. Astrom, K. J., and P. Eykhoff, System identification—A survey, Automatica 7(2), 123-162 (March 1971).
5. Astrom, K. J., and B. Wittenmark, On self-tuning regulators, Automatica 9(2), 185-199 (March 1973).
6. Aubin, B., Strategies for mechanizing structural induction, in Proceedings of the 5th International Joint Conference on Artificial Intelligence, Vol. 1, M.I.T., Cambridge, Massachusetts, August 1977, pp. 363-369.
7. Banerji, R., Learning to solve games and puzzles, in Computer Oriented Learning Processes (J. C. Simon, ed.), Noordhoff, Leyden, 1976, pp. 341-368.
8. Barrow, H. G., and R. J. Popplestone, Relational descriptions in picture processing, in Machine Intelligence, Vol. 7 (B. Meltzer and D. Michie, eds.), American Elsevier, New York, 1972, pp. 377-396.
9. Bauer, M., A basis for the acquisition of procedures from protocols, in Proceedings of the 4th International Joint Conference on Artificial Intelligence, Vol. 1, M.I.T., Cambridge, Massachusetts, September 1975, pp. 226-231.
10. Bierman, A. W., and J. A. Feldman, A survey of results in grammatical inference, in Frontiers of Pattern Recognition (S. Watanabe, ed.), Academic, New York, 1972, pp. 31-54.
11. Bruner, J. S., J. J. Goodnow, and G. A. Austin, A Study of Thinking, Wiley, New York, 1956.
12. Buchanan, B. G., Scientific theory formation by computer, in Computer Oriented Learning Processes (J. C. Simon, ed.), Noordhoff, Leyden, 1976, pp. 515-534.
13. Buchanan, B. G., and T. M. Mitchell, Model-directed learning of production rules, in Pattern-Directed Inference Systems (D. A. Waterman and F. Hayes-Roth, eds.), Academic, New York, forthcoming.

14. Chen, C. H., Statistical pattern—Review and outlook, IEEE Syst., Man Cybern. Newsl. 6(4), 7-8 (August 1977).
15. Churchman, C. W., The role of Weltanschauung in problem solving and inquiry, in Theoretical Approaches to Non-Numerical Problem Solving (R. B. Banerji and M. D. Mesarovic, eds.), Springer, New York, 1970, pp. 141-151.
16. Cook, C. M., and A. Rosenfeld, Some experiments in grammatical inference, in Computer Oriented Learning Processes (J. C. Simon, ed.), Noordhoff, Leyden, 1976, pp. 157-174.
17. Davis, R., Applications of Meta-Level Knowledge to the Construction, Maintenance, and Use of Large Knowledge Bases, STAN-CS-76-552, Computer Science Department, Stanford University, July 1976.
18. Donalson, D. D., and F. H. Kishi, Review of adaptive control theories and techniques, in Modern Control Systems Theory (C. T. Leondes, ed.), McGraw-Hill, New York, 1965, pp. 228-284.
19. Duda, R. O., and P. E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
20. Elcock, E. W., and A. M. Murray, Experiments with a learning component in a go-moku playing program, in Machine Intelligence 1 (Collins and Michie, eds.), Oliver & Boyd, London, 1967, pp. 87-103.
21. Erman, L. D., R. D. Fennell, V. R. Lesser, and D. R. Reddy, System organization for speech understanding, in International Joint Conference on Artificial Intelligence-3, Advance Papers, Stanford, California, August 1973, pp. 194-199.
22. Evans, T. G., A program for the solution of a class of geometric analogy intelligence test questions, in Semantic Information Processing (M. Minsky, ed.), M.I.T. Press, Cambridge, Massachusetts, 1968, pp. 271-353.
23. Feigenbaum, E. A., The simulation of verbal learning behaviour, in Computers and Thought (E. A. Feigenbaum and J. Feldman, eds.), McGraw-Hill, New York, 1963, pp. 297-309.
24. Feigenbaum, E. A., B. G. Buchanan, and J. Lederberg, On generality and problem solving: A case study using the DENDRAL program, in Machine Intelligence 6 (B. Meltzer and D. Michie, eds.), American Elsevier, New York, 1971, pp. 165-190.
25. Fel'dbaum, A. A., Dual control theory I-IV, Autom. Remote Control 21(9), 874-880 (April 1961); 21(11), 1033-1039 (June 1961); 22(1), 1-12 (August 1961); 22(2), 109-121 (September 1961). Also collected in R. Oldenburger (ed.), Optimal and Self-Optimizing Control, M.I.T. Press, Cambridge, Massachusetts, 1966, pp. 458-496.
26. Fikes, R., P. Hart, and N. J. Nilsson, Learning and executing generalized robot plans, Artif. Intell. 3, 251-288 (1972).
27. Findler, N. V., and W. R. McKinsie, Computer simulation of a self-preserving and learning organism, Bull. Math. Biophys. 31, 247-253 (1969).
28. Findler, N. V., Studies in machine cognition using the game of poker, Commun. ACM 20(4), 230-245 (1977).
29. Friedberg, R. M., A learning machine: Part 1, IBM J. 2, 2-13 (January 1958)
30. Fu, K. S., Learning control systems—Review and outlook, IEEE Trans. Autom. Control 15(2), 210-221 (April 1970).
31. Fu, K. S., Syntactic Methods in Pattern Recognition, Academic, New York, 1974.

32. Fu, K. S., A brief review of pattern recognition, IEEE Syst., Man Cybern. Newsl. 6(4), 3-5 (August 1977).
33. Fu, K. S., and T. L. Booth, Grammatical inference: Introduction and survey– Part I, IEEE Trans. Syst., Man Cybern. SMC-5(1), 95-111 (1975); Part II, IEEE Trans. Syst., Man Cybern. SMC-5(4), 409-423 (July 1975).
34. Glorioso, R. M., Engineering Cybernetics, Prentice-Hall, Englewood Cliffs, New Jersey, 1975.
35. Gold, E. M., Language identification in the limit, Inf. Control 10, 447-474 (1967).
36. Green, C. C., The design of the PSI program synthesis system, in Proceedings of the Second International Conference on Software Engineering, San Francisco, October 1976, pp. 4-18.
37. Griffith, A. K., A comparison and evaluation of three machine learning procedures as applied to the game of checkers, Artif. Intell. 5, 137-148 (1974).
38. Hajek, P., On logics of discovery, in Lecture Notes in Computer Science, Vol. 32 (J. Becvar, ed.), Springer, 1975, pp. 30-45.
39. Hardy, S., Synthesis of LISP functions from examples, in Proceedings of the 4th International Joint Conference on Artificial Intelligence, Vol. 1, M. I. T., Cambridge, Massachusetts, September 1975, pp. 240-245.
40. Hasdorff, L., Gradient Optimization and Nonlinear Control, Wiley, New York, 1976.
41. Hayes-Roth, F., and J. McDermott, Knowledge acquisition from structural descriptions, in Proceedings of the 5th International Joint Conference on Artificial Intelligence, Vol. 1, M. I. T., Cambridge, Massachusetts, August 1977, pp. 356-362.
42. Hayes-Roth, F., and D. Mostow, An automatically compilable recognition network for structured patterns, in Proceedings of the 4th International Joint Conference on Artificial Intelligence, Vol. 1, M.I.T., Cambridge, Massachusetts, September 1975, pp. 246-251.
43. Hedrick, C., Learning production-systems from examples, Artif. Intell. 7, 21-49 (1976).
44. Holland, J. H., Outline for a logical theory of adaptive systems, in Essays on Cellular Automata (A. W. Burks, ed.), University of Illinois Press, Chicago, 1970.
45. Hunt, E. B., Artificial Intelligence, Academic, New York, 1975.
46. Hunt, E. B., and C. I. Hovland, Programming a model of human concept formation, in Computers and Thought (E. Feigenbaum and J. Feldman, eds.), McGraw-Hill, New York, 1963, pp. 310-325.
47. Hunt, E. B., J. Marin, and P. T. Stone, Experiments in Induction, Academic, New York, 1966.
48. Johnson, C. R., Jr., A new procedure for adaptive control design, in Proceedings of the 11th Asilomar Conference on Circuits, Systems, and Computers, Pacific Grove, California, November 1977, pp. 130-133.
49. Johnson, D. L., and D. C. Holden, Computer learning in theorem proving, IEEE Trans. Syst., Sci. Cybern. SSC-2, 115-123 (1966).
50. Johnson, E. S., An information processing model of one kind of problem solving, Psychol. Monogr. Whole No. 581 (1964).
51. Kalman, R. E., Design of a self-optimizing control system, Trans. ASME 80(2), 468-478 (1958). Also in R. Oldenburger (ed.), Optimal and Self-Optimizing Control, M.I.T. Press, Cambridge, Massachusetts, 1966, pp. 440-449.
52. Kanal, L., Patterns in pattern recognition: 1968-1974, IEEE Trans. Inf. Theory IT-20(6), 697-722 (November 1974).

53. Kanal, L., Current status, problem and prospects of pattern recognition, IEEE Syst., Man Cybern. Newsl. 6(4), 9-11 (August 1977).
54. Koffman, E. B., Learning games through pattern recognition, IEEE Trans. Syst., Sci. Cybern. SSC-4, 12-16 (1968).
55. Koford, J. S., and G. F. Groner, The use of an adaption threshold elements to design a linear optimal pattern classifier, IEEE Trans. Inf. Theory 12(1), 42-50 (1966).
56. Kuhn, T. S., The Structure of Scientific Revolutions, rev. ed., University of Chicago Press, Chicago, 1970.
57. Landau, I. D., A survey of model reference adaptive techniques—Theory and applications, Automatica 10(4), 353-379 (July 1974).
58. Landau, I. D., and B. Courtiol, Design and multivariable adaptive model following control systems, Automatica 10(5), 483-494 (September 1974).
59. Langley, P. W., BACON: A production system that discovers- empirical laws, in Proceedings of the 5th International Joint Conference on Artificial Intelligence, Vol. 1, M. I. T., Cambridge, Massachusetts, August 1977, pp. 344-346.
60. Larson, J., and R. S. Michalski, Inductive inference of VL decision rules, in Pattern-Directed Inference Systems (D. A. Waterman and F. Hayes-Roth, eds.), Academic, New York, forthcoming.
61. Lenat, D. B., AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search, Ph.D. Thesis, Stanford University, 1976. See also Automated theory formation in mathematics, in Proceedings of the 5th International Joint Conference on Artificial Intelligence, Vol. 2, M. I. T., Cambridge, Massachusetts, August 1977, pp. 833-841.
62. Lesser, V. R., R. D. Fennell, L. D. Erman, and D. R. Reddy, Organization of the HEARSAY II speech understanding system, IEEE Trans. Acoust., Speech, Signal Process. ASSP-23(1), 11-23 (February 1975).
63. Lindorff, D. P., and R. L. Carroll, Survey of adaptive control using Liapunov design, Int. J. Control 18(5), 897-914 (November 1973).
64. McCarthy, J., Programs with common sense, in Semantic Information Processing (M. Minsky, ed.), M.I.T. Press, Cambridge, Massachusetts, 1968, pp. 403-418.
65. McCarthy, J., Book review of J. Lighthill AI Investigation, Artif. Intell. 5(3), 317-322 (Fall 1974).
66. Meltzer, B., Generation of hypotheses and theories, Nature 225, 972 (March 7, 1970).
67. Meltzer, B., The programming of deduction and induction, in Artificial and Human Thinking (A. Elithorn and D. Jones, eds.), San Francisco, 1973, pp. 19-33.
68. Mendel, J. M., and K. S. Fu (eds.), Adaptive, Learning and Pattern Recognition Systems: Theory and Applications, Academic, New York, 1970.
69. Michie, D., On Machine Intelligence, Wiley, New York, 1974.
70. Minsky, M., Steps toward artificial intelligence, in Computers and Thought (E. A. Feigenbaum and J. Feldman, eds.), McGraw-Hill, New York, 1963, pp. 406-450.
71. Minsky, M., and S. Papert, Perceptrons, M.I.T. Press, Cambridge, Massachusetts, 1969.
72. Minsky, M., and S. Papert, Artificial Intelligence—Progress Report, A.I. MIT AI Memo 252, M. I. T., Cambridge, Massachusetts, January 1972.
73. Mitchell, T. M., Version spaces: A candidate elimination approach to rule learning, in Proceedings of the 5th International Joint Conference on Artificial Intelligence, Vol. 1, M. I. T., Cambridge, Massachusetts, August 1977, pp. 305-310.

74. Monopoli, R. V., Model reference adaptive control with an augmented error signal, IEEE Trans. Autom. Control AC-19(5), 474-484 (October 1974).
75. Narendra, K. S., and M. A. L. Thathachar, Learning automata—A survey, IEEE Trans. Syst., Man Cybern. SMC-4(4), 323-333 (July 1974).
76. Newell, A., Learning, generality and problem solving, in Proceedings of the YFIP Congress 62 (C. M. Popplewell, ed.), North Holland, Amsterdam, 1963, pp. 407-412.
77. Newell, A., Artificial intelligence and the concept of mind, in Computer Models of Thought and Language (R. Schank and K. Colby, eds.), Freeman, San Francisco, 1973, pp. 1-60.
78. Newman, C., and L. Uhr, BOGART: A discovery and induction program for games, in Proceedings of ACM 20th National Conference (L. Winner, ed.), Winner, New York, pp. 176-185.
79. Nilsson, N. J., Learning Machines, McGraw-Hill, New York, 1965.
80. Oettinger, A. G., Programming a digital computer to learn, Philos. Mag. 43, 1243-1263 (1952).
81. Pavlidis, T., Comments on current perspectives in pattern recognition, IEEE Syst., Man Cybern. Newsl. 6(4), 8-9 (August 1977).
82. Persson, S., Some Sequence Extrapolating Problems, Ph.D. Thesis, University of California, Berkeley, 1966 (see also Stanford AI Memo No. 44).
83. Pitrat, J., Realization of a program learning to find combinations at chess, in Computer Oriented Learning Processes (J. C. Simon, ed.), Noordhoff, Leyden, 1976, pp. 397-423.
84. Plotkin, G. D., A further note on inductive generalization, in Machine Intelligence, Vol. 6 (B. Meltzer and D. Michie, eds.), Edinburgh University Press, Edinburgh, 1971, pp. 101-124. Machine Intelligence, Vol. 5
85. Popplestone, R. J., An experiment in automatic induction, in Machine Intelligence, Vol. 5 (B. Meltzer and D. Michie, eds.), Edinburgh University Press, Edinburgh, 1970, pp. 203-215.
86. Proceedings of International Joint Conference on Pattern Recognition (First: Washington, D.C., October 1973; Second: Copenhagen, Denmark, August 1974; Third: Coronado, California, November 1976).
87. Reddy, D. R., Speech Recognition, Academic, New York, 1975.
88. Reddy, D. R., L. D. Erman and R. B. Neily, A model and a system for machine recognition of speech, IEEE Trans. Audio Electroacoust. AII-21(3), 229-238 (June 1973).
89. Roche, C., Application of multilevel clustering to the automatic generation of recognition operators. A link between feature extraction and classification, in Second International Joint Conference on Pattern Recognition, Copenhagen, August 1974, pp. 540-546.
90. Rosenblatt, F., The pereeptron, a probabilistic model for information organization and storage in the brain, Psychol. Rev. 65, 368-408 (1958).
91. Rovner, P., B. Nash-Webber, and W. A. Woods, Control concepts in a speech understanding system, IEEE Trans. Acoust., Speech, Signal Process. ASSP-23(1), 136-140 (February 1975).
92. Rychener, M. D., and A. Newell, An instructable production system: Basic design issues, in Pattern-Directed Inference Systems (D. A. Waterman and F. Hayes-Roth, eds.), Academic, New York, forthcoming.
93. Samuel, A. L., Some studies in machine learning using the game of checkers, in Computers and Thought (E. A. Feigenbaum and J. Feldman, eds.), McGraw-Hill, New York, 1963, pp. 71-105.

94. Samuel, A. L., Some studies in machine learning using the game of checkers II—Recent progress, IBM J. Res. Dev. 11(6), 601-617 (November 1967).
95. Saridis, G. N., and R. N. Lobbia, Parameter identification and control of linear discrete-time systems, IEEE Trans. Autom. Control AC-17(1), 52-60 (February 1972). Comments on "Parameter identification and control of linear discrete-time systems, " IEEE Trans. Autom. Control AC-20(3), 442-443 (June 1975).
96. Selfridge, O. G., Pandemonium: A paradigm for learning, in Proceedings of the Symposium of Mechanization of Thought Processes, HMSO, London, 1959, pp. 511-529.
97. Selfridge, O. G., and U. Neisser, Pattern recognition by machine, in Computers and Thought (E. A. Feigenbaum and J. Feldman, eds.), McGraw-Hill, New York, 1963, pp. 237-268.
98. Shackcloth, B., and R. L. Butchart, Synthesis of model reference adaptive systems by Liapunov's second method, in Theory of Self-Adaptive Control Systems (P. H. Hammond, ed.), Plenum, New York, 1966, pp. 145-152.
99. Shaw, D., W. Swartout, and C. Green, Inferring LISP programs from examples, in Proceedings of the 4th International Joint Conference on Artificial Intelligence, Vol. 1, M.I.T., Cambridge, Massachusetts, 1975, pp. 260-267.
100. Siklossy, L., Natural language learning by computer, in Representation and Meaning (H. A. Simon and L. Siklossy, eds.), Prentice-Hall, Englewood Cliffs, New Jersey, 1972, pp. 288-328.
101. Siklossy, L., Procedural learning in worlds of robots, in Computer Oriented Learning Processes (J. C. Simon, ed.), Noordhoff, Leyden, 1976, pp. 427-440.
102. Simon, H. A., Scientific discovery and the psychology of problem solving, in Mind and Cosmos (R. G. Colodny, ed.), University of Pittsburgh Press, Pittsburgh, 1966, pp. 22-40.
103. Simon, H. A., The Sciences of the Artificial, M.I.T. Press, Cambridge, Massachusetts, 1969.
104. Simon, H. A., and K. Kotovsky, Human acquisition of concepts for sequential patterns, Psyched. Rev. 70, 534-546 (1963).
105. Simon, H. A., and G. Lea, Problem solving and rule induction: A unified view, in Knowledge and Cognition (L. W. Gregg, ed.), Lawrence Erlbaum Associates, Potomac, Maryland, 1974, pp. 105-127.
106. Sklansky, J., Adaptation, learning, self-repair, and feedback, IEEE Spectrum 1(5), 172-174 (1964).
107. Smith, M. H., A learning program which plays partnership dominoes, Commun. ACM 16, 462-467 (August 1973).
108. Solomonoff, R., Inductive inference theory—A unified approach to problems in pattern recognition and artificial intelligence, in Proceedings of the 4th International Joint Conference on Artificial Intelligence, Vol. 1, M. I. T., Cambridge, Massachusetts, 1975, pp. 274-280.
109. Soloway, E. M., and E. M. Riseman, Knowledge-directed learning, in Pattern-Directed Inference Systems (D. A. Waterman and F. Hayes-Roth, eds.), Academic, New York, forthcoming.
110. Soloway, E. M., and E. M. Riseman, Levels of pattern description in learning, in Proceedings of the 5th International Joint Conference on Artificial Intelligence, Vol. 2, M.I.T., Cambridge, Massachusetts, 1977, pp. 801-811.

111. Stockman, G. C., A Problem-Reduction Approach to the Linguistic Analysis of Waveforms, TR-538, Department of Computer Science, University of Maryland, May 1977.
112. Sussman, G. J., A Computational Model of Skill Acquisition, MIT AI-TR-297, Al Laboratory, M. I. T., August 1973.
113. Tse, E., and Y. Bar-Shalom, An actively adaptive control for linear systems with random parameters via the dual control approach, IEEE Trans. Autom. Control AC-18(2), 109-117 (April 1973).
114. Tse, E., and Y. Bar-Shalom, Actively adaptive control for nonlinear stochastic systems, Proc. IEEE 64(7), 1172-1181 (August 1976).
115. Tsypkin, Ya. Z., Self learning—What is it?, IEEE Trans. Autom. Control AC-13(6), 608-612 (December 1968).
116. Uhr, L., Pattern Recognition, Learning and Thought, Prentice-Hall, Englewood Cliffs, New Jersey, 1973.
117. Uhr, L., and C. Vossler, A pattern-recognition program that generates, evaluates, and adjusts its own operators, in Computers and Thought (E. A. Feigenbaum and J. Feldman, eds.), McGraw-Hill, New York, 1963, pp. 251-268.
118. Vere, S. A., Inductive learning of relational productions, in Pattern-Directed Inference Systems (D. A. Waterman and F. Hayes-Roth, eds.), Academic, New York, forthcoming.
119. Wald, A., Sequential-Analysis, Wiley, New York, 1947.
120. Watanabe, S. (ed.), Methodologies of Pattern Recognition, Academic, New York, 1969.
121. Waterman, D. A., Generalization learning techniques for automating the learning of heuristics, Artif. Intell. 1(1-2), 121-170 (1970).
122. Waterman, D. A., Adaptive production systems, in Proceedings of the 4th International Joint Conference on Artificial Intelligence, M. I. T., Cambridge, Massachusetts, 1975, pp. 296-303.
123. Waterman, D. A., Exemplary programming in RITA, in Pattern-Directed Inference Systems (D. A. Waterman and F. Hayes-Roth, eds.), Academic, New York, forthcoming.
124. White, G. M., Machine Learning through Signature Trees. Application to Human Speech, AI Memo 136 (CS-183), Stanford AI Laboratory, Stanford University, October 1970 (AD-717 600).
125. Widrow, B., The “rubber-mask” technique—H. Pattern storage and recognition, Pattern Recognition 5, 199-211 (1973).
126. Widrow, B., and M. E. Hoff, Adaptive switching circuits, 1960 IRE WESCON Conv. Record, Part 4, 96-104 (August 1960).
127. Wiener, N., Cybernetics, Wiley, New York, 1948.
128. Winston, P. H., Learning Structural Descriptions from Examples, MIT AI-TR-231, AI Laboratory, M.I.T., September 1970.
129. Winston, P. H. (ed.), The Psychology of Computer Vision, McGraw-Hill, New York, 1975.
130. Wittenmark, B., Stochastic adaptive control methods: A survey, Int. J. Control 21(5), 705-730 (1975).
131. Zadeh, L. A., Outline of a new approach to the analysis of complex systems and decision processes, IEEE Trans. Syst., Man Cybern. SMC-3, 28-44 (January 1973).

132. Zagoruiko, N., Empirical prediction algorithms, in Computer Oriented Learning Processes (J. C. Simon, ed.), Noordhoff, Leyden, 1976, pp. 581-595.

Bruce G. Buchanan
Tom M. Mitchell
Reid G. Smith
C. Richard Johnson Jr.