Stanford Heuristic Programming Project
Memo HPP-77-39

January 1979

Computer Science Department
Report No. STAN-CS-79-692



# MODELS OF LEARNING SYSTEMS

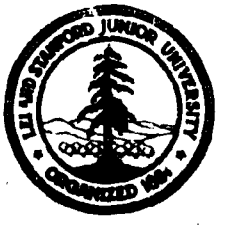
by

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



COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY





UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: HPP-77-39
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): MODELS OF LEARNING SYSTEMS
5. TYPE OF REPORT & PERIOD COVERED: Technical Report
6. PERFORMING ORG. REPORT NUMBER: HPP-77-39
7. AUTHOR(s): Bruce G. Buchanan, Tom M. Mitchell, Reid G. Smith and C. Richard Johnson, Jr.
8. CONTRACT OR GRANT NUMBER(s): MDA 903-77-C-0270, PHS-RR-00612
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Computer Science Department, Stanford University, Stanford, California 94305
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency, Information Processing Techniques Office, 1400 Wilson Avd., Arlington, VA 22209
12. REPORT DATE: January 1979
13. NUMBER OF PAGES: 38
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Mr. Philip Surra, Resident Representative, Office of Naval Research, Durand 165, Stanford University
15. SECURITY CLASS. (of this report): Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report): Releasable without limitations on dissemination.

DISTRIBUTION STATEMENT A
Approved for public release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report): STAN-CS-79-692, HPP-77-39
18. SUPPLEMENTARY NOTES
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The terms adaptation, learning, concept-formation, induction, self-organization, and self-repair have all been used in the context of learning system (LS) research. In this article, three distinct approaches to machine learning and adaptation are considered: (i) the adaptive control approach, (ii) the pattern recognition approach, and (iii) the artificial intelligence approach.

Progress in each of these areas is summarized in the first part of the,

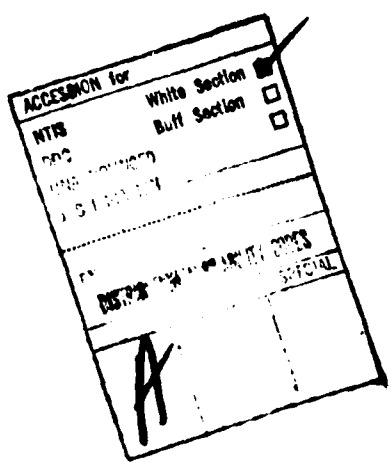
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

094 120

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Article. In the next part a general model for learning systems is presented that allows characterization and comparison of individual algorithms and programs in all of these areas. The model details the functional components felt to be essential for any learning system, independent of the techniques used for its construction, and the specific environment in which it operates. Specific examples of learning systems are described in terms of the model.



# MODELS OF LEARNING SYSTEMS

Bruce G. Buchanan, Tom M. Mitchell, and Reid G. Smith[1]

Departments of Computer Science and Electrical Engineering
Stanford University
Stanford, California, 94305.

C. Richard Johnson Jr.

Department of Electrical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, Virginia, 24601.

## Abstract

The terms adaptation, learning, concept-formation, induction, self-organization, and self-repair have all been used in the context of learning system (LS) research. In this article, three distinct approaches to machine learning and adaptation are considered: *(i)* the adaptive control approach, *(ii)* the pattern recognition approach, and *(iii)* 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 that allows characterization and comparison of individual algorithms and programs in all of these areas. The model details the functional components felt to be essential for any learning system, independent of the techniques used for its construction, and the specific environment in which it operates. Specific examples of learning systems are described in terms of the model.

[1] To appear in J. Belzer (Ed.), *Encyclopedia Of Computer Science And Technology*, Marcel Dekker, Inc., New York, 1978, Vol. 11. This research has been supported in part by the National Institutes of Health Grant No. 5R24 RR 00612-09; Advanced Research Projects Agency Grant No. MDA 903-77-C-02777; and the Department of National Defence of Canada.

## 1 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 [Wiener, 1948], [Ashby, 1956]. 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 that can learn to play good checkers [Samuel, 1963], [Samuel, 1967], learn to filter out the strong heartbeat of a mother in order to pick out the weaker heartbeat of the fetus [Widrow, 1973], or learn to predict the mass spectra of complex molecules [Buchanan, 1978]. 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 that 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 that uses information obtained during one interaction with its environment to improve its performance during future interactions. This rough characterization may include man/machine systems (see [McCarthy, 1968]) 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.

Figure 1 shows several classes of systems that fit the above characterization and lists the kinds of alterations that they perform. Data base systems are among the earliest kinds of systems that fit our definition. Such systems represent information about their environment by sets of alterable assertions. In the late 1950's and early 1960's, adaptive control techniques were first used to build programs that alter parameters in equations which model some aspect of the external world [Samuel, 1963], [Widrow, 1973]. The perceptrons of the early 1960's [Minsky, 1972], [Rosenblatt, 1958] 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 LSs is their internal representations of the outer environment: some are mathematical models, some are linguistic assertions, and still others are structures encoding symbolic relations.

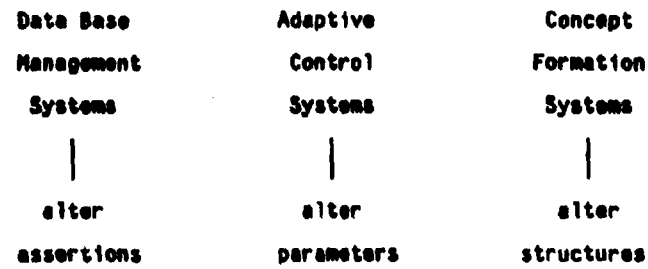



**Figure 1. A Spectrum Of Learning Systems.**

In this article, three distinct approaches to machine learning and adaptation are considered: *(i)* the adaptive control approach, *(ii)* the pattern recognition approach, and *(iii)* 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 that 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.

## 2 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 that has been chosen by the LS designer to represent the system under study [Donalson, 1966], [Fu, 1970]. 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 [Sklansky, 1964]. 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 [Wittenmark, 1975].

There are many overlapping and sometimes contradictory definitions of the terms related to adaptive systems. The following set, formulated by Glorioso [Glorioso, 1975], serves to illustrate the main features. An *adaptive system* is defined as a system that 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-1950's [Asher, 1976]. 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 [Tse, 1973]. *Active adaptive controllers* are based on dual control theory [Fel'dbaum, 1966]. In addition to the available real-time information, they utilize the knowledge that future observations will be made that 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.

## 2.1 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 [Kalman, 1958], 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 [Astrom, 1971]. 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 [Saridis, 1975].

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 [Hsaderff, 1976] or heuristic [Fu, 1970] 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 [Linderff, 1973]. The earliest application of Liapunov function synthesis for designing adaptive loops [Shackcloth, 1965] utilized a model reference approach.

*Model reference* adaptive control techniques (see example in 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 [Landau, 1974a], [Monopoli, 1974] by occasionally necessitating an unbounded control effort. Control effort boundedness is encouraged by abandoning exact output matching for input matching [Johnson, 1976], 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.

## 3 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 non-human 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 that 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.

### 3.1 Pattern Recognition Subclasses

Pattern recognition is presently characterized by two major approaches: 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 [Roche, 1974]. 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 that may characterize patterns. [Kanal, 1974].

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 [Reddy, 1973], [Erman, 1973], [Lesser, 1975], [Rovner, 1975], [Reddy, 1975], 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 artificial intelligence approaches to learning system design. We have recently begun to see a merger of the two approaches (see, for example Stockman, [Stockman, 1977]), that may result in more powerful systems. For a review of the current state of the art, see [Chen, 1977], [Pavlidis, 1977], [Kanal, 1977] and [Proceedings, 1976].

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 towards exploration of the description model.

### 3.1.1 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 a set of probability density functions, or *fuzzy*, wherein patterns are thought to have differing degrees of membership in a number of classes [Zadeh, 1973].

#### *Variations*

Classifiers may be categorized in a number of ways, depending on the type of classification rule, and the sampling procedure they employ [Hunt, 1975].

*Parallel* classifiers base their classifications upon the complete set of features, extracted simultaneously during a single observation of a pattern. *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 [Wald, 1947].

*Adaptive* classifiers (see example in 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 [Widrow, 1960]). *Non-adaptive* 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 it 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 non-parametric 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 [Duda, 1973].

### 3.1.2 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 [Duda, 1973], [McCarthy, 1974] where classification is clearly inappropriate. It also tends to be useful when the patterns themselves are complex [Fu, 1977], as it emphasizes hierarchical decomposition of patterns into their constituent components.

There are a variety of descriptive formalisms in which to express the structural descriptions. These include pattern grammars [Fu, 1974], and relational graphs [Winston, 1975]. *Pattern grammars* embody an attempt to carry over a large amount of theory from the study of natural and programming languages. A variety of pattern grammars have been developed [Kanal, 1974], 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 [Winston, 1975]). 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 [Duda, 1973] is an excellent introduction to the methods used in the structural approach.

## 4 Artificial Intelligence Approach to Learning

In the 1950's and early 1960's there was considerable discussion of learning programs in the Artificial Intelligence (*AI*) literature (e.g., [Oettinger, 1952], [Friedberg, 1958], [Selfridge, 1959], [Newell, 1962], [Feigenbaum, 1963], [Minsky, 1963] and [Simon, 1966]). 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 [Minsky, 1972]. The 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 [Samuel, 1963] 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 [Newell, 1973]:

> Inductive tasks have always been a prominent part of the *AI* 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* [Bruner, 1956], 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.,[Simon, 1963] [Hunt, 1963], [Feigenbaum, 1963] and [Hunt, 1966]), others are written to demonstrate the *feasibility of a method* (e.g., [Moloway, 1978]), and still others are written with the express purpose of *aiding human problem solvers* codify and explain data (e.g., [Buchanan, 1976]). 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 [Selfridge, 1959], [Uhr, 1963] and [Uhr, 1973]). Much work has been performed on learning automata [Nilsson, 1965] (see also [Narendra, 1974]), and neural networks that grow in response to stimuli [Minsky, 1972]. All of these efforts have aimed at defining simple machines that learn to respond to their environments [Fiedler, 1966]. Newell [Newell, 1973] traces one line of growth from stimulus-response learning in

psychology to *(i)* pattern recognition and self-organizing systems, as well as to *(ii)* concept formation, induction and other *AI* work. The two fields diverged in the 1960's, 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 [Feigenbaum, 1963] 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 [Fikes, 1972], [Sussman, 1973], and [Winston, 1975] 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 [Feigenbaum, 1971], [McCarthy, 1968] and [Minsky, 1972a]. 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 [Davis, 1976].

Various levels of sophistication in learning systems are described by [Winston, 1975]: 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., [Solomonoff, 1977], [Holland, 1962], [Hajek, 1975], [Meltzer, 1970], [Meltzer, 1973] and [Plotkin, 1971]). 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 [Buchanan, 1978] 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 [Larson, 1978] 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 [Vere, 1978], is able to learn rules for *(i)* extending letter sequences, *(ii)* recognizing geometric analogies, *(iii)* relating *before and after* situations, and *(iv)* relating sequences of situations. It uses background knowledge about the domain to help it recognize important relations among features of objects that are not codified in the descriptions of the objects themselves.

### 4.1 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, [Newman, 1966]). 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 [Samuel, 1963], [Samuel, 1967]. 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. [White, 1970]). (Griffith [Griffith, 1974] later compared the methods used by Samuel with a simple heuristic procedure.) Waterman [Waterman, 1970] 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, that the learning system altered in light of mistakes. Waterman has generalized many of these ideas to other tasks [Waterman, 1975]. Findler [Findler, 1977] has also studied the game of poker. Pitrat's work on learning patterns in chess [Pitrat, 1974] applies many heuristic search ideas to learning useful combinations from examples of given games. Programs have also been written to learn dominoes [Smith, 1973], Go-Moku [Elcock, 1967], and the rules of Tic-Tac-Toe [Popplestone, 1969]. Banerji [Banerji, 1974] has studied learning processes for several classes of games and puzzles from a more formal point of view. Koffman [Koffman, 1968] has also related game playing to pattern recognition.

### 4.2 Concept Formation

In concept formation tasks, a computer program (or human subject) is presented with objects, or descriptions of objects, that exhibit a common concept. The program (or subject) is expected to generalize from these instances well enough to classify new objects accurately. Negative instances--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 [Simon, 1963]. Or they have been written to learn by 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 [Evans, 1968] and Winston [Winston, 1975]. 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 D1, 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 tees and posts with a fallen lintel.

Another recent program learns concepts, such as Hit and Out, for the game of baseball from a set of descriptions of events over the span of a game [Soloway, 1978]. Other concept formation programs are described in [Simon, 1963], [Johnson, 1964], [Hunt, 1975], [Zagoruiko, 1976], [Langley, 1977], [Larson, 1978], [Buchanan, 1978], [Mitchell, 1977], [Hayes-Roth, 1975], [Hayes-Roth, 1977], [Hedrick, 1976] and [Rychener, 1978].

### 4.3 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 for 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 -> 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 that 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 [Bierman, 1972], [Persson, 1966] 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 [Cook, 1976]), but most recent work on grammatical inference emphasizes more formal methods [Biermann, 1972], [Gold, 1967] and [Fu, 1975].

Inferring natural language [Siklossy, 1972], and simple computer programs from examples are other induction tasks that have been studied using *AI* techniques [Waterman, 1975], [Hardy, 1975], [Shaw, 1977], [Waterman, 1978]. 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.

## 5 A Model of Learning Systems

This section is concerned with a simple functional model that is useful for characterizing, comparing, and designing learning systems. Many of the functional components of an LS are essential to intelligent problem solving systems in general, as noted by Simon and Lea [Simon, 1973]; 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.

### 5.1 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 [Samuel, 1963], [Samuel, 1967], and grammatical inference programs [Hunt, 1975].

*Unsupervised* LSs operate within an environment of instances for which the correct response is not directly available. The version of Samuel's program that learns by playing checkers against an opponent falls into this category [Samuel, 1963] 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 [Minsky, 1963]. Tsypkin [Tsypkin, 1968] has pointed out that unsupervised learning is somewhat of an illusion in the sense that a teacher/designer defines the standards that 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 [Winston, 1975], 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 [Buchanan, 1978].) ***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 [Barrow, 1972], [Duda, 1973], [Donalson, 1966].

### 5.2 The Model - Overview

The proposed LS model is shown in Figure 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 that specify functions that must be performed to effect learning. 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.

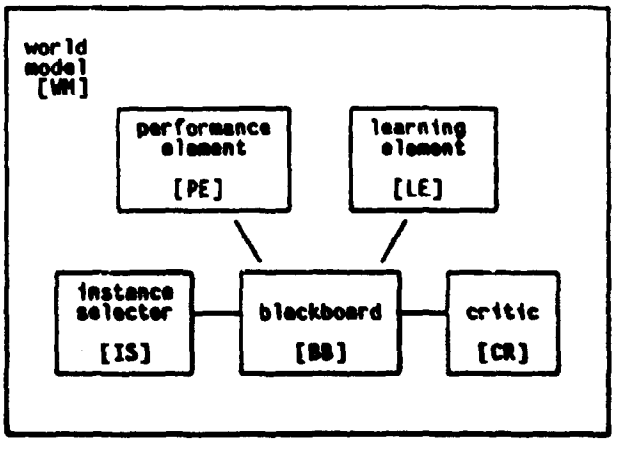



**Figure 2. The Components of a Learning System**

The following sections present detailed discussions of the LS model components shown in Figure 2. In addition, the appendix contains detailed characterizations of representative *AI*, 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.

### 5.3 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 [Waterman, 1970], [Waterman, 1975] have the potential to exhibit richer variations than systems whose operations are keyed only to the adjustment of parameter values [Landau, 1974], [Michie, 1974].

### 5.4 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 LSs, 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 [Samuel, 1963] is externally controlled, whereas Popplestone's program [Popplestone, 1968], 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 [Simon, 1975].) In the adaptive systems literature, Tse and Bar-Shalom [Tse, 1975] 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 [Winston, 1975] 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 [Donalson, 1965].

### 5.5 Critic

The critic analyses 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 [Fu, 1970], [Glorioso, 1975], [Sklansky, 1964]. Feedback from a critic at least as evaluator is essential for learning.

The critic may also localize errors and localize the reasons for poor localize the reasons for poor performance. This type of behavior is essential for resolution of the credit assignment problem described by Minsky [Minsky, 1963]. In its diagnostic role, the critic is exemplified by the bug classifier and summarizer in Sussman's HACKER [Sussman, 1973].

Finally the critic may recommend repairs by making specific recommendations for improvement or suggestions about future instances. In Waterman's poker player [Waterman, 1970], the critic in this 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 LSs 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 LSs the functions of the critic have been left to humans. For example, **MYCIN/TEIRESIAS** [Davis, 1976] 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 semi-automated.

### 5.6 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 [Waterman, 1970], parameterized polynomials [Samuel, 1963], executable procedures [Sussman, 1975], signature tables [Samuel, 1967], stored facts [Feigenbaum, 1963], and graphs or networks [Winston, 1975]. The method of incorporating new learned information is dependent upon the representation, and even among systems that use similar representations, competing methods are found (contrast, for example, [Buchanan, 1978] and [Waterman, 1970]).

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 [Winston, 1975]. 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 that 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 that 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 LSs fall into this category [Nilsson, 1965]. Other systems consider several training instances at a time in order to minimize the effect of occasional noisy instances [Buchanan, 1978].

### 5.7 Blackboard

The blackboard of this model is a global data base that also functions as a system communications mechanism. It is similar to the concept introduced in the HEARSAY system [Lesser, 1975]. 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.

### 6.8 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 [Churchman, 1970]. 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 [Winston, 1975]). 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 LSs. 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.

### 6.9 Multi-Layer 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 Figure 2. This leads us to the concept of layered LSs, 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, that has its own functional components (critic, world model, etc.), as shown in Figure 8. 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 inter-layer 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 multi-layer context.

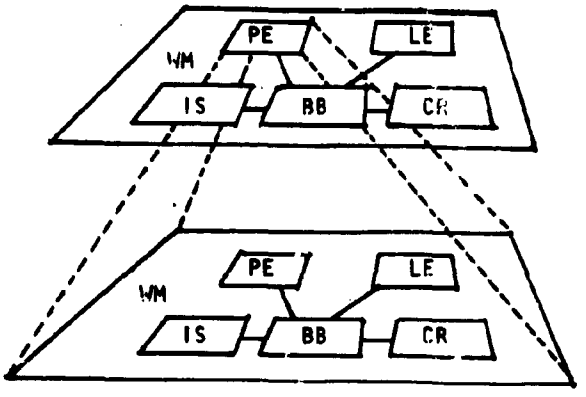



**Figure 3. Layering of Learning Systems. (Components are labelled as in Figure 2).**

This multi-layer 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 [Simon, 1969] and includes as a specific case the bottom-to-top hierarchical architecture used, for example, by Soloway [Soloway, 1977].

One existing LS which may be viewed as a layered system is the version of Samuel's program [Samuel, 1967] that 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 [Kuhn, 1970]. However, a shift in the conceptual framework of LS1 could be made by a properly programmed LS2 [Buchanan, 1974]. 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 [Samuel, 1963], [Uhr, 1963] and [Soloway, 1977], 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.

### 5.10 Implications of the Model

The LS model described here provides a common language for characterization and comparison of different types of learning systems that 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 multi-layer 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 A

## Characterization of Existing Systems

In this appendix several existing LSs are characterized using the framework provided by the model described in Section 5. 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*, [Landau, 1974]

**Purpose:** Construct a *controller* that 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--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 that 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*, [Koford, 1966]

**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 pre-specified set of classes. Each pattern is represented as a feature vector.

**Performance Element:** A linear pattern classifier that forms the inner product of a pattern feature vector (that 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 [Widrow, 1960], 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 formulation 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*, [Samuel, 1963], [Samuel, 1967]

**Purpose:** Learn to play good game of checkers (here we discuss only the version of the program that 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 pre-defined game-board/recommended-move pairs.

**Critic:** Evaluation--examines the ranking given to the book move by the performance element. Localization--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--analyses the learning ability of LS1 (i.e., the LS2 performance element) with the current set of evaluation function features. Localization--singles out features that are not useful. Recommendation--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 that may be considered, and the performance standard employed by the LS2 critic.

*Poker Player*, [Waterman, 1970]

**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. Localization--gives important state variables for deciding the correct bet. Recommendation--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 recommendation are deduced from the next action of the opponent and a set of predefined axioms, while localization 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, [Buchanan, 1978]**

**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. Localization--states whether the rule is acceptable, too specific, or too general. Recommendation--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*, [Winston, 1970], [Winston, 1975]**

**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. Localization--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.

## References

[Arkadev, 1971]
A. G. Arkadev and E. M. Braverman, *Learning in Pattern Classification Machines*. Nauka, Moscow, 1971.

[Ashby, 1956]
W. R. Ashby, *An Introduction to Cybernetics*. Wiley, New York, 1963 (originally published, 1956).

[Asher, 1976]
R. B. Asher, D. Andrisani, II, and P. Dorato, Bibliography on adaptive control systems. *Proc. IEEE*, 64 (8), 1226-1240 (1976).

[Astrom, 1971]
K. J. Astrom and P. Eykhoff, System identification--a survey. *Automatica*, 7 (2), 123-162 (1971).

[Astrom, 1973]
K. J. Astrom and B. Wittenmark, On self-tuning regulato[illegible] (2), 185-199 (1973).

[Aubin, 1977]
R. Aubin, Strategies for Mechanizing Structural Inducti[illegible] *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridg[illegible], August 1977, Vol 1:363-369.

[Banerji, 1974]
R. Banerji, Learning to Solve Games and Puzzles. In *Computer Oriented Learning Processes*, (J. C. Simon, Ed.), Noordhoff, Leyden, 1976.

[Barrow, 1972]
H. G. Barrow and R. J. Popplestone, Relational Descriptions in Picture Processing. In *Machine Intelligence 7*, (B. Meltzer and D. Michie, Eds.), American Elsevier, New York, 1972, pp. 377-396.

[Bauer, 1975]
M. Bauer, A Basis for the Acquisition of Procedures From Protocols. *Proceedings of the 4th IJCAI*, Cambridge, Mass., September 1975, Vol. 1:226-231.

[Bierman, 1972]
A. W. Bierman and J. A. Feldman, A Survey of Results in Grammatical Inference. In *Frontiers of Pattern Recognition*, (S. Watanabe, Ed.), Academic Press, New York, 1972, pp.31-54.

[Bruner, 1956]
J. S. Bruner, J. J. Goodnow, and G. A. Austin, *A Study of Thinking*. Wiley, New York, 1956.

[Buchanan, 1974]
B. G. Buchanan, Scientific Theory Formation by Computer. In *Computer Oriented Learning Processes*, (J.C. Simon, Ed.), Noordhoff, Leyden, 1976.

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

[Chen, 1977]
C. H. Chen, Statistical Pattern-Review and Outlook. *IEEE SMC Newsletter*, 6(4), 7-8, (1977).

[Churchman, 1970]
C. W. Churchman, 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-Verlag, New York, 1970, pp. 141-151.

[Cook, 1976]
C. M. Cook and A. Rosenfeld, Some Experiments in Grammatical Inference. In *Computer Oriented Learning Processes*, (J. C. Simon, Ed.), Noordhoff, Leyden, 1976.

[Davis, 1976]
R. Davis, Applications of Meta-Level Knowledge to the Construction, Maintenance, and Use of Large Knowledge Bases. STAN-CS-76-552, Stanford University, July 1976.

[Donalson, 1965]
D. D. Donalson 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.

[Duda, 1973]
R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[Elcock, 1967]
E. W. Elcock 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.

[Erman, 1973]
L. D. Erman, 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, Calif., August 1973, pp. 194-199.

[Evans, 1968]
T. G. Evans, A Program for the Solution of a Class of Geometric Analogy Intelligence Test Questions. In *Semantic Information Processing* (M. Minsky, Ed.), MIT Press, Cambridge, Mass., 1968, pp. 271-283.

[Feigenbaum, 1963]
E. A. Feigenbaum, The Simulation of Verbal Learning Behavior. In *Computers and Thought* (E. A. Feigenbaum and J. Feldman, Eds.), New York: McGraw-Hill, 1963, pp. 297-309.

[Feigenbaum, 1971]
E. A. Feigenbaum, 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.

[Fel'dbaum, 1965]
A. A. Fel'dbaum, Dual control theory I-IV. In *Optimal and Self-Optimizing Control* (R. Oldenburger, Ed.), MIT Press, Cambridge, Mass., 1965, pp. 458-496. (Originally published, *Autom. Remote Control 21*(9) 874-880 (1961); *21*(11) 1033-1039 (1961); *22*(1) 1-12 (1961); *22*(2) 109-121 (1961)).

[Fikes, 1972]
R. Fikes, P. Hart, and N. J. Nilsson, Learning and Executing Generalized Robot Plans. *Artificial Intelligence*, 3, 251-288 (1972).

[Findler, 1969]
N. V. Findler and W. R. McKinsie, Computer Simulation of a Self-Preserving and Learning Organism. *Bull. Math. Biophysics, 31*, 247-263 (1969)

[Findler, 1977]
N. V. Findler, Studies in Machine Cognition Using the Game of Poker. *CACM*, *20*(4), 230-245 (1977).

[Friedberg, 1958]
R. M. Friedberg, A Learning Machine: Part 1. *IBM Journal*, *2*, 2-13 (1958).

[Fu, 1970]
K. S. Fu, Learning Control Systems - Review and Outlook. *IEEE Trans. on Automatic Control*, *15*(2) 210-221 (1970).

[Fu, 1974]
K. S. Fu, *Syntactic Methods in Pattern Recognition*. Academic Press, New York, 1974.

[Fu, 1975]
K. S. Fu and T. L. Booth, Grammatical Inference: Introduction and Survey-Part I. *IEEE Trans. on SMC*, *SMC-5*(1), 95-111 (1975); Part II, *IEEE Trans. on SMC*, *SMC-5*(4) 409-423 (1975).

[Fu, 1977]
K. S. Fu, A Brief Review of Pattern Recognition. *IEEE SMC Newsletter*, *6*(4) 3-5 (1977).

[Glorioso, 1975]
R. M. Glorioso, *Engineering Cybernetics*. Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

[Gold, 1967]
E. M. Gold, Language Identification in the Limit. *Information and Control, 10*, 447-474, (1967).

[Green, 1976]
C. C. Green, The Design of the PSI Program Synthesis System. *Proceedings of the Second International Conference on Software Engineering*, San Francisco, California, October 1976, pp. 4-18.

[Griffith, 1974]
A. K. Griffith, A Comparison And Evaluation Of Three Machine Learning Procedures As Applied To The Game Of Checkers. *Artificial Intelligence 5*, 137-148 (1974).

[Hajek, 1976]
P. Hajek, On Logics of Discovery. In *Lecture Notes in Computer Science* (J. Becvar, Ed.), Springer-Verlag, New York, 1976, Vol. 1:30-45.

[Hardy, 1975]
S. Hardy, Synthesis of LISP Functions from Examples. *Proceedings of the 4th IJCAI Conference*, Cambridge, Mass. September 1975, Vol 1:240-245.

[Hasdorff, 1976]
L. Hasdorff, *Gradient Optimization and Nonlinear Control*. John Wiley and Sons, New York, 1976.

[Hayes-Roth, 1975]
F. Hayes-Roth and D Mostow, An Automatically Compilable Recognition Network For Structured Patterns. *Proceedings of the 4th IJCAI Conference*, Cambridge, Mass., September 1975, Vol. 1:356-362.

[Hayes-Roth, 1977]
F. Hayes-Roth and J. McDermott, Knowledge Acquisition from Structural Descriptions. *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge, Mass., August 1977, Vol.1:248-251.

[Hedrick, 1976]
C. Hedrick, Learning Production-Systems from Examples. *Artificial Intelligence, 7*, 21-49 (1976).

[Holland, 1962]
J. H. Holland, Outline for a Logical Theory of Adaptive Systems. In *Essays on Cellular Automata* (A.W. Burks, Ed.). Univ. of Illinois Press, Chicago, 1970.

[Hunt, 1975]
E. B. Hunt, *Artificial Intelligence*. Academic Press, New York, 1975.

[Hunt, 1963]
E. B. Hunt 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.

[Hunt, 1966]
E. B. Hunt, J. Marin, and P. T. Stone, *Experiments in Induction*. Academic Press, New York, 1966.

[Johnson, 1978]
C. R. Johnson, Jr., Input matching: a new procedure for adaptive control design. *Proceedings of the 11th Asilomar Conference on Circuits, Systems and Computers*, Pacific Grove, Calif., November 1977, pp. 130-133.

[Johnson, 1966]
D. L. Johnson and D. C. Holden, Computer Learning in Theorem Proving. *IEEE Transactions on Systems Science and Cybernetics*, *SSC-2*, 115-123 (1966).

[Johnson, 1964]
E. S. Johnson, An Information Processing Model of One Kind of Problem Solving. *Psychol. Monographs*, Whole No. *581* (1964).

[Kalman, 1966]
R. E. Kalman, Design of a self-optimizing control system. In *Optimal and Self-Optimizing Control* (R. Oldenburger, Ed.), MIT Press, Cambridge, Mass., 1966, pp. 440-449, (originally published, 1958).

[Kanal, 1974]
L. Kanal, Patterns in Pattern Recognition: 1968-1974. *IEEE Trans on Inform. Theory*, *IT-20*(6), 697-722 (1974).

[Kanal, 1977]
L. Kanal, Current Status, Problem and Prospects of Pattern Recognition. *IEEE SMC Newsletter*, *6*(4), 9-11 (1977).

[Koffman, 1968]
E. B. Koffman, Learning games through pattern recognition. *IEEE Transactions on Systems Science and Cybernetics*. *SSC-4*, 12-16 (1968).

[Koford, 1966]
J. S. Koford and G. F. Groner, The Use of an Adaption Threshold Elements to Design a Linear Optimal Pattern Classifier. *IEEE Trans. on Information Theory*, *12*(1) 42-50 (1966).

[Kuhn, 1970]
T. S. Kuhn, *The Structure of Scientific Revolutions*. 2nd ed., University of Chicago Press, Chicago, 1970.

[Landau, 1974]
I. D. Landau, A Survey of Model Reference Adaptive Techniques - Theory and Applications. *Automatica*, *10*(4) 353-379 (1974)

[Landau, 1974a]
I. D. Landau and B. Courtiol, Design and multivariable adaptive model following control systems. *Automatica*, *10*(5) 483-494 (1974).

[Langley, 1977]
P. W. Langley, BACON: A Production System That Discovers Empirical Laws. *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge, Mass, August 1977, Vol 1:344-346.

[Larson, 1978]
J. Larson and R. S. Michalski, Inductive Inference of VL Decision Rules. In *Pattern-Directed Inference Systems* (D. A. Waterman and F. Hayes-Roth, Eds.), Academic Press, New York, 1978.

[Lenat, 1976]
D. B. Lenat, *AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search*, PhD Thesis, Stanford University, Stanford, California, 1976. See also *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge, Mass., August 1977, Vol.2:833-841.

[Lesser, 1975]
V. R. Lesser, R. D. Fennell, L. D. Erman, and D. R. Reddy, Organization of the HEARSAY II Speech Understanding System. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, *ASSP-23*(1) 11-23 (1975).

[Lindorff, 1973]
D. P. Lindorff and R. L. Carroll, Survey of adaptive control using Liapunov design. *Int. J. Control*, *18*(5) 897-914(1973).

[McCarthy, 1968]
J. McCarthy, Programs with Common Sense. In *Semantic Information Processing*. MIT Press, Cambridge, Mass., 1968, pp. 403-418.

[McCarthy, 1974]
J. McCarthy, Book Review of J. Lighthill AI Investigation. *Artificial Intelligence*, *5*(3), 317-322 (1974).

[Meltzer, 1970]
B. Meltzer, Generation of Hypotheses and Theories. *Nature*, */2268*, 972 (March 1970).

[Meltzer, 1973]
B. Meltzer, The programming of Deduction and Induction. In *Artificial and Human Thinking*, (A. Elithorn and D. Jones, Eds.), San Francisco, 1973, pp. 19-33.

[Mendel, 1970]
J. M. Mendel and K. S. Fu, (eds.), *Adaptive, Learning and Pattern Recognition Systems: Theory and Applications*, New York: Academic Press, 1970.

[Michie, 1974]
D. Michie, *On Machine Intelligence*, Wiley, New York, 1974.

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

[Minsky, 1972]
M. Minsky and S. Papert, *Perceptrons*, The MIT Press, Cambridge, Mass., 1969.

[Minsky, 1972a]
M. Minsky and S. Papert, *Artificial Intelligence - Progress Report*, A.I. MIT AI Memo 252, January 1972.

[Mitchell, 1977]
T. M. Mitchell, Version Spaces: A Candidate Elimination Approach to Rule Learning. *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, MIT, Cambridge, Mass., August 1977, Vol 1: 305-310.

[Monopoli, 1974]
R. V. Monopoli, Model reference adaptive control with an augmented error signal, *IEEE Trans. on Aut. Control, AC-19*(5), 474-484 (1974).

[Narendra, 1974]
K. S. Narendra and M. A. L. Thathachar, Learning Automata-A Survey. *IEEE Trans. Syst., Man Cybern., SMC-4*(4), 323-333 (1974).

[Newell, 1962]
A. Newell, Learning, Generality and Problem Solving. In *Proceedings of the IFIP Congress, 62*, Amsterdam, North-Holland, 1963, pp. 407-412.

[Newell, 1973]
A. Newell, Artificial Intelligence and the Concept of Mind. *Computer Models of Thought and Language* (R. Schank and K. Colby, Eds.), W. H. Freeman, San Francisco, 1973, pp. 1-60.

[Newman, 1965]
C. Newman and L. Uhr, BOGART: A Discovery and Induction Program for Games. *Proceedings of ACM 20th National Conference*, 1965, pp. 176-185.

[Nilsson, 1965]
N. J. Nilsson, *Learning Machines*, McGraw-Hill, New York, 1965.

[Oettinger, 1952]
A. G. Oettinger, Programming a Digital Computer to Learn. *Phil. Mag., 43*, 1243-1263 (1952).

[Pavlidis, 1977]
T. Pavlidis, Comments on Current Perspectives in Pattern Recognition, *IEEE Syst., Man Cybern. Newsl. 6*(4), 6-9 (1977).

[Persson, 1966]
S. Persson, *Some Sequence Extrapolating Problems.* (PhD Thesis, University of California, Berkeley). Stanford Artificial Intelligence Memo No. 44, 1966.

[Pitrat, 1974]
J. Pitrat, Realization of a Program Learning to Find Combinations at Chess. *Computer Oriented Learning Processes* (J.C. Simon, Ed.), Noordhoff, Leyden, 1976, pp. 397-423.

[Plotkin, 1971]
G.D. Plotkin, A Further Note on Inductive Generalization, *Machine Intelligence 6* (B. Meltzer and D. Michie, Eds.), Edinburgh University Press, Edinburgh, 1969, pp. 101-124.

[Popplestone, 1969]
R. J. Popplestone, An Experiment in Automatic Induction. *Machine Intelligence - 5* (B. Meltzer and D. Michie, Eds.), Edinburgh University Press, 1970, pp. 204-215.

[Proceedings, 1976]
Proceedings of Int. Joint Conf. Pattern Recognition (First: Washington, D.C., October 1973; Second: Copenhagen, Denmark, August, 1974; Third: Coronado, California, November, 1976.)

[Reddy, 1973]
D. R. Reddy, L. D. Erman and R. B. Neely, A Model and a System for Machine Recognition of Speech. *IEEE Trans. on Audio and Electroacoustics, AU-21*(3), 229-238, (1973).

[Reddy, 1975]
D. R. Reddy, *Speech Recognition*, Academic Press, New York, 1975.

[Roche, 1974]
C. Roche, Application of Multilevel Clustering to the Automatic Generation of Recognition Operators. A Link Between Feature Extraction and Classification. *Presented at the Second International Joint Conference on Pattern Recognition*, Copenhagen, August, 1974, pp. 540-546.

[Rosenblatt, 1958]
F. Rosenblatt, The Perceptron, A Probabilistic Model for Information Organization and Storage in the Brain. *Psychological Review, 65*, 386-408 (1958).

[Rovner, 1975]
P. Rovner, B. Nash-Webber and W. A Woods, Control Concepts in a Speech Understanding System, *IEEE Trans. on ASSP*, *ASSP-23*,(1), 136-140 (1975).

[Rychener, 1978]
M. D. Rychener and A. Newell, An Instructable Production System: Basic Design Issues. *Pattern-Directed Inference Systems* (D.A. Waterman and F. Hayes-Roth, Eds.), Academic Press, New York, 1978.

[Samuel, 1963]
A. L. Samuel, Some Studies in Machine Learning Using the Game of Checkers. *Computers and Though:* (E.A. Feigenbaum and J. Feldman, Eds.), McGraw-Hill, New York, 1963, pp. 71-105.

[Samuel, 1967]
A. L. Samuel, Some Studies in Machine Learning Using the Game of Checkers II - Recent Progress. *IBM Journal of Research and Development*, *11*(6), 601-617, (1967).

[Saridis, 1975]
G. N. Saridis and R. N. Lobbia, Parameter identification and control of linear discrete-time systems, *IEEE Trans. on Aut. Control*, *AC-17*(1) 52-60 (1972); *Comments on* Parameter identification and control of linear discrete-time systems. *IEEE Trans. on Aut. Control*, *AC-20*(3), 442-443(1975).

[Selfridge, 1959]
O. G. Selfridge, Pandemonium: A Paradigm for Learning. *Proceedings of the Symposium on Mechanization of Thought Processes* (D. Blake and A. Uttley, Eds.), HMSO, London, 1959, pp. 511-529.

[Selfridge, 1963]
O. G. Selfridge and U. Neisser, Pattern Recognition by Machine, Pattern recognition by machine,. *Computers and Thought* (E.A. Feigenbaum and J. Feldman, Eds.), McGraw-Hill, New York, 1963, pp. 237-268.

[Shackcloth, 1966]
B. Shackcloth and R. L. Butchart, Synthesis of model reference adaptive systems by Liapunov's second method. *Theory of Self-Adaptive Control Systems*, (P. H. Hammond, Ed.), Plenum Press, New York, 1966, pp. 145-152.

[Shaw, 1977]
D. Shaw, W. Swartout, and C. Green, Inferring LISP Programs From Examples. *Proceedings of the 4th IJCAI*, MIT, Cambridge, Mass., 1975, Vol. 1: 260:267.

[Siklossy, 1972]
L. Siklossy, Natural language learning by computer. *Representation of Meaning* (H.A. Simon and L. Siklossy, Eds.), Prentice-Hall, Englewood Cliffs, New Jersey, 1972, pp. 288-328.

[Siklossy, 1974]
L. Siklossy, Procedural learning in worlds of robots. *Computer Oriented Learning Processes* (J.C. Simon, Ed.), Noordhoff, Leyden, 1976, pp. 427-440.

[Simon, 1963]
H. A. Simon and K. Kotovsky, Human Acquisition of Concepts for sequential patterns. *Psychol. Rev. 70*, 534-546 (1963).

[Simon, 1966]
H.A. Simon, Scientific discovery and the psychology of problem solving, *Mind and Cosmos* (R.G. Colodny, Ed.), Univ. of Pittsburgh Press, Pittsburgh, 1966, pp. 22-40.

[Simon, 1969]
H.A. Simon, *The Sciences of the Artificial*, MIT Press, Cambridge, Mass., 1969.

[Simon, 1973]
H. A. Simon and G. Lea, Problem Solving and Rule Induction: A unified view. *Knowledge and Cognition* (L.W. Gregg, Ed.), Lawrence Erlbaum Associates, Potomac, Maryland, 1974, pp. 105-127.

[Sklansky, 1964]
J. Sklansky, Adaptation, Learning, Self-Repair, and feedback, *IEEE Spectrum 1*(5), 172-174 (1964).

[Smith, 1973]
M. H. Smith, A Learning Program Which Plays Partnership dominoes. *Commun. ACM, 16*, 462-467 (August 1973).

[Solomonoff, 1977]
R. Solomonoff, Inductive Inference Theory - A Unified Approach To Problems in Pattern Recognition and Artificial Intelligence. *Proceedings of the 4th IJCAI*, MIT, Cambridge, Mass., 1975, Vol.1:274-280.

[Soloway, 1977]
E. M. Soloway and E. M. Riseman, Levels of Pattern description in learning. *Proceedings of the 5th IJCAI*, MIT, Cambridge, Mass., 1977, Vol.2: 801-811.

[Soloway, 1978]
E. M. Soloway and E. M. Riseman, Knowledge-Directed learning. *Pattern-Directed Inference Systems* (D.A. Waterman and F. Hayes-Roth, Eds.), Academic Press, New York, 1978.

[Stockman, 1977]
G. C. Stockman, A Problem-Reduction Approach to the Linguistic Analysis of Waveforms. TR-538, Dept. of Computer Science, University of Maryland, May 1977.

[Sussman, 1975]
G. J. Sussman, A Computational Model of Skill Acquisition. MIT AI-TR-297, August 1975.

[Tse, 1973]
E. Tse and Y. Bar-Shalom, An actively adaptive control for linear systems with random parameters via the dual control approach. *IEEE Trans. on Aut. Control AC-13*(6), 608-612 (December 1968).

[Tse, 1976]
E. Tse and Y. Bar-Shalom, Actively Adaptive Control for Nonlinear Stochastic Systems, *Proc. IEEE 64*(7), 1172-1181 (August 1976).

[Tsypkin, 1968]
Ya. Z. Tsypkin, Self Learning - What is it?, *IEEE Trans. on Automatic Control, AC-18*(2), 109-117 (1973).

[Uhr, 1963]
L. Uhr and C. Vossler, A Pattern-Recognition Program That generates, evaluates, and adjusts its own operators. *Computers and Thought* (E.A. Feigenbaum and J. Feldman, Eds.), McGraw-Hill, New York, 1963, pp. 251-268.

[Uhr, 1973]
L. Uhr, *Pattern Recognition, Learning and Thought*, Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

[Vere, 1978]
S. A. Vere, Inductive Learning of Relational Productions. *Pattern-Directed Inference Systems* (D.A. Waterman and F. Hayes-Roth, Eds.), Academic Press, New York, 1978.

[Wald, 1947]
Wald, A., *Sequential Analysis*, Wiley, New York, 1947.

[Watanabe, 1969]
S. Watanabe (Ed.), *Methodologies of Pattern Recognition*. Academic Press, New York, 1969.

[Waterman, 1970]
D. A. Waterman, Generalization Learning Techniques for automating the learning of heuristics, *Artif. Intell. 1*(1,2), 121-170 (1970).

[Waterman, 1975]
D. A. Waterman, Adaptive Production Systems. *Proceedings of the 4th IJCAI*, MIT, Cambridge, Mass., 1975, pp.296-303.

[Waterman, 1978]
D. A. Waterman, Exemplary Programming in RITA. *Pattern-Directed Inference Systems* (D.A. Waterman and F. Hayes-Roth, Eds.), Academic Press, New York, 1978.

[White, 1970]
G. M. White, Machine Learning Through Signature Trees. Application to Human Speech. AI Memo 136 (CS-183) Stanford AI Laboratory, October 1970 (AD-717 600).

[Widrow, 1960]
B. Widrow and M. E. Hoff, Adaptive Switching Circuits, *1960 IRE WESCON Conv. Record*, Part 4,96-104 (August 1960).

[Widrow, 1973]
B. Widrow, The 'Rubber-Mask' Technique - II. Pattern Storage and Recognition, *Pattern Recognition*, *5*, 199-211 (1973).

[Wiener, 1948]
N. Wiener, *Cybernetics*, Wiley, New York, 1948.

[Winston, 1970]
P. H. Winston, Learning Structural Descriptions From Examples, MIT AI-TR-231, September 1970.

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

[Wittenmark, 1975]
B. Wittenmark, Stochastic adaptive control methods: a survey, *Int. J. Control 21* (5), 705-730 (1975).

[Zadeh, 1973]
L. A. Zadeh, 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).

[Zagoruiko, 1976]
N. Zagoruiko, Empirical Prediction Algorithms, *Computer Oriented Learning Processes* (J.C. Simon, Ed.), Noordhoff, Leyden, 1976.