Learning Control Systems—Review and Outlook

KING-SUN FU, SENIOR MEMBER, IEEE

Abstract—**The basic concept of learning control is introduced. The following five learning schemes are briefly reviewed: 1) trainable controllers using pattern classifiers, 2) reinforcement learning control systems, 3) Bayesian estimation, 4) stochastic approximation, and 5) stochastic automata models. Potential applications and problems for further research in learning control are outlined.**

I. INTRODUCTION

IN designing an optimal control system, if all the *a priori* information about the controlled process (plant environment) is known and can be described deterministically, the optimal controller is usually designed by deterministic optimization techniques. If all or a part of the *a priori* information can be described only statistically, for example, in terms of probability distribution or density functions, then stochastic or statistical design techniques are used. However, if the *a priori* information required is unknown or incompletely known, in general, an optimal design cannot be achieved by classical methods of optimal design (dynamic programming, maximum principle, variational calculus, etc.). Two different approaches have been taken to solve this class of problems. One approach is to design a controller based only upon the amount of information available. In that case, the unknown information is either ignored or is assumed to take some known values from the designer's best guess. The second approach is to design a controller which is capable of estimating the unknown information during its operation and an optimal control action is determined on the basis of the estimated information. In the first case, a rather conservative design criterion (for example, minimax criterion) is often used; the systems designed are, in general, inefficient and suboptimal. In the second case, if the estimated information gradually approaches the true information as time proceeds, then the controller thus designed will approach the optimal controller. In other words, the performance of the designed controller will eventually be as good as in the case where all the *a priori* information required is known. Because of the gradual improvement of performance due to the improvement of the estimated unknown information, this class of control systems may be called learning control systems [1]–[11]. The controller learns the unknown information during operation and the learned information is, in turn, used as an experience for future decisions or controls.[1]

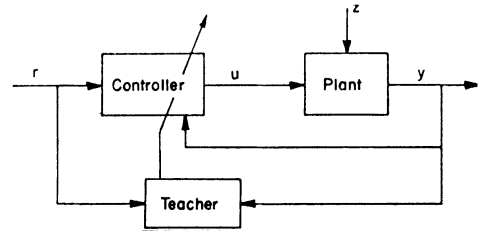

Fig. 1. Learning control system.

From the concept just introduced, the problem of learning may be viewed as the problem of estimation or successive approximation of the unknown quantities of a functional which represent the controlled process under study. The unknown quantities to be estimated or learned by the controller may be either the parameters only, or the form and parameters which describe a deterministic or probabilistic function. The relationship between the control law and this function is usually chosen by the designer (for example, in terms of a preselected optimization criterion). Therefore, as the controller accumulates more information about the unknown function or parameters, the control law will be altered according to the updated information in order to improve the system's performance. A basic block diagram for a learning control system is shown in Fig. 1. The dynamics of the plant under the environmental disturbance Z are assumed unknown or partially known. Therefore, there is a need to design a controller which will learn (or estimate) the unknown information required for an optimal control law. The actual control action is determined on the basis of the learned (or the estimated) information and is, in general, suboptimal. However, if the learned information converges to the true information as time proceeds, the sub-

Manuscript received February 1, 1969; revised July 25, 1969. This work was supported by the National Science Foundation under Grant GK-1970 and by the U.S. Air Force Office of Scientific Research under Grant 69-1776.

The author is with Purdue University, Lafayette, Ind. 47907.

Reprinted from IEEE TRANSACTIONS ON AUTOMATIC CONTROL, vol. AC-15, no. 2, April 1970.

[1]The use of the word "adaptive" has been intentionally avoided here. Both adaptive and learning are used to describe behavior or performance of systems (including engineering and biological systems). The term "learning" seems relatively easier to be explained in terms of the appropriate utilization of past experience and the gradual improvement of performance. From the conventional notion of adaptation (although still controversial), every learning system is also adaptive. However, the contrary may not be true for many so-called adaptive systems. On the other hand, the term "self-organizing" has been used more or less to describe some internal structure change capability of systems. Although the structure change of a system usually results in behavior change or improvement, the term "self-organizing system" has only provided a very fuzzy description of a system's external behavior. It may be considered that adaptive and learning are behavior-descriptive terms, but feedback and self-organizing are structure- or system configuration-descriptive terms. Nevertheless, the terminology war is still going on. It is certainly not the purpose of this paper to get involved in such a war.

optimal controller is expected to approach to the optimal controller asymptotically. The "teacher" evaluates the performance of the controller and directs the learning process performed by the controller so the system's overall performance will be gradually improved.

Depending upon whether or not an external supervision (in the form of a teacher) is required, the process of learning can be classified into 1) learning with external supervision (or training or supervised or off-line learning), and 2) learning without external supervision (or nonsupervised or on-line learning). In learning processes with external supervision, the desired answer (for example, the desired output of the system or the desired optimal control action) is usually considered exactly known. Directed by the known answer (given by an external teacher, say), the controller modifies its control strategy or control parameters to improve the system's performance. On the other hand, in learning processes without external supervision, the desired answer is not exactly known. Two approaches are usually employed in designing learning controllers. The first approach is that the learning process is carried out by considering all possible answers (the mixture approach in Bayesian learning) [47]–[52]. The second approach is that the controller uses a performance measure to direct the learning process (performance feedback approach) [33], [56], [67]. The learned information is considered as an experience of the controller, and the experience will be used to improve the quality of control whenever similar control situations recur. The new information extracted from a recurred control situation is used to update the estimation or the experience associated with that control situation. Different experiences are obtained from the information extracted from different control situations. Similar control situations may be grouped to form a class of control situations. A major function also performed by some learning controllers is the classification of different classes of control situations such that an optimal control law can be gradually established for the various classes of control situations and the admissible control actions.

II. Pattern Classifcation

Since the problem of classifying different classes of control situations is important to the design of a learning controller, the general problem of pattern classification is briefly introduced in this section. Suppose that a set of measurements or observations is taken to represent an unknown pattern or a control situation. These measurements (called features) are designated as $x_1, x_2, \cdots, x_k$, and can be represented by a k-dimensional vector X in the (feature) space Ω_X. Let the m possible pattern classes (or m classes of control situations) be $\omega_1, \omega_2, \cdots, \omega_m$. The function of a pattern classifier is to assign (or to make a decision about) the correct class membership to each given feature vector X. Such an operation can be interpreted as a partition of the k-dimensional space Ω_x into m mutually exclusive regions (or a mapping from the space Ω_x to the decision space). The partitioning boundary or decision

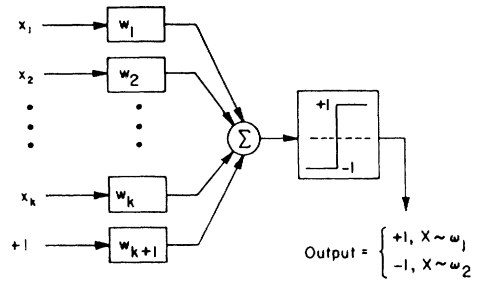

Fig. 2. Linear two-class classifier.

surface can be expressed in terms of discriminant functions. Associated with each ω_i, a discriminant function $d_i(X)$, $i = 1, \cdots, m$ is selected such that if X is from class ω_i then

$$d_i(X) > d_j(X), \quad \text{for all } j \neq i. \tag{1}$$

The decision surface between the class ω_i and the class ω_j is represented by the equation

$$d_i(X) - d_j(X) = 0. \tag{2}$$

There are many ways for selecting $d_i(X)$. Several important discriminant functions are discussed in the following [14].

A. Linear Discriminant Function

The discriminant function $d_i(X)$ is selected as a linear function of feature measurements $x_1, x_2, \cdots, x_k$; i.e.,

$$d_i(X) = \sum_{r=1}^{k} w_{ir} x_r + w_{i,k+1}, \quad i = 1, \cdots, m. \tag{3}$$

The decision surface represented by

$$f(X) = d_i(X) - d_j(X) = \sum_{r=1}^{k} (w_{ir} - w_{jr})\, x_r + (w_{i,k+1} - w_{j,k+1}) = 0 \tag{4}$$

is also a linear function of x_r or, in other words, a hyperplane in the space Ω_X. Let

$$w_r = w_{ir} - w_{jr}, \quad r = 1, \cdots, k+1$$

then (4) becomes

$$\sum_{r=1}^{k} w_r x_r + w_{k+1} = 0. \tag{5}$$

For $m = 2$, a two-class linear classifier can be easily implemented by a threshold logic device shown in Fig. 2. If the input feature X is from ω_1, i.e., $X \sim \omega_1$, then the output of the threshold logic device will be $+1$ since

$$d_1(X) - d_2(X) = \sum_{r=1}^{k} w_r x_r + w_{k+1} > 0.$$

On the other hand, if

$$\sum_{r=1}^{k} w_r x_r + w_{k+1} < 0$$

then the output will be -1 and $X \sim \omega_2$. For $m > 2$,

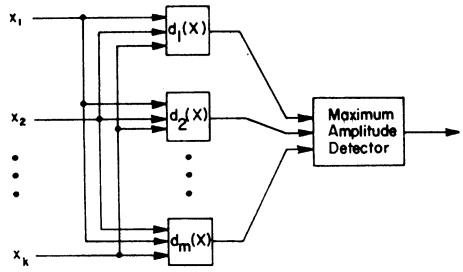

Fig. 3. m-class classifier.

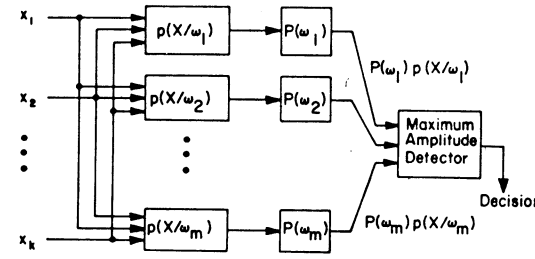

Fig. 4. Bayes classifer.

several threshold logic devices connected in parallel can be used for classification purposes. The various combinations of $+1$ and -1 at the outputs of each threshold logic device will give different classifications. In general, using Fig. 2, an m-class classifier can be implemented as shown in Fig. 3.

B. Polynomial Discriminant Function

The discriminant function is selected as an nth order ($n > 1$) polynomial of $x_1, x_2, \cdots, x_k$. In particular, if $n = 2$,

$$d_i(X) = \sum_{r=1}^{k} w_{rr}x_r^2 + \sum_{r=1}^{k-1}\sum_{q=r+1}^{k} w_{rq}x_r x_q + \sum_{r=1}^{k} w_r x_r + w_{N+1} \tag{6}$$

where $N = k + k(k-1)/2 + k = k(k+3)/2$. Let $A = [a_{ij}]$ where $a_{jj} = w_{jj}, j = 1, \cdots, k$,

$$a_{jq} = \tfrac{1}{2}w_{jq}, \quad j, q = 1, \cdots, k, \quad j \neq q$$

and let B be a column vector with element $b_j = w_j, j = 1, \cdots, k$. Then (6) can be written in vector matrix form

$$d_i(X) = X^T AX + X^T B + C \tag{7}$$

where X^T is the transpose of X and $C = w_{N+1}$. The decision surface between ω_i and ω_j is, in general, a hyper-hyperboloid. In some special cases, the decision surface may be hypersphere or hyperellipsoid.

In a more general formulation, the decision surface $f(X) = 0$ between ω_i and ω_j in the feature space can be expressed as [23]-[26]

$$f(X) = \sum_{l=1}^{N} c_l\varphi_l(X) = 0 \tag{8}$$

where $\varphi_l(X)$ is a complete set of functions defined[2] on Ω_X and c_l the coefficients in the expansions.[3]

C. Statistical Discriminant Function

The discriminant functions selected in the first two cases are usually assumed functions of the deterministic vector variable X. However, if the noise contaminating the feature measurements and the variations of all patterns in each class are considered, X is usually assumed to be a vector-valued random variable. In such a case, one may select a discriminant function of the following form:

$$d_i(X) = P_i p(X|\omega_i), \quad i = 1, \cdots, m \tag{9}$$

where P_i is the *a priori* probability of class ω_i and $p(X|\omega_i)$ is a multivariate conditional probability density function of X given $X \sim \omega_i$. The decision rule for classifying pattern classes using (9) as the discriminant function corresponds to the Bayes' decision rule with zero-one loss function in the statistical decision theory [19]. A block diagram for this type of pattern classifier is shown in Fig. 4.

If the cost of taking feature measurements is to be considered or the features measured are sequential in nature, one is led to use a sequential decision approach [20], [21]. In this case, the feature measurements are taken in sequence. After each measurement, the classifier makes a decision either to terminate the process and make a terminal decision about the class membership, or to take an additional measurement. The error probability (probability of misrecognition) can be prespecified and the number of feature measurements required for a terminal decision is not fixed but a random variable. The advantage of using a sequential decision approach is that, on the average, the number of feature measurements is less than that required in a nonsequential case for the same error probability. For example, in a two-class classification problem, Wald's sequential probability ratio test can be applied [20]. After each feature measurement is taken, compare the sequential probability ratio

$$\lambda_k = p_k(X|\omega_1)/p_k(X|\omega_2), \quad k = 1, 2, \cdots \tag{10}$$

with two stopping bounds A and B, where $p_k(X|\omega_i), i = 1, 2$ is the conditional density function of X given $X \sim \omega_i$ after k measurements have been taken. The stopping bounds A and B are related to the probability of misrecognition with the following relationship:

$$A = (1 - \epsilon_{21})/\epsilon_{12}, \quad B = \epsilon_{21}/(1 - \epsilon_{12})$$

where ϵ_{12} is the probability of classifying X as in ω_1 when actually $X \sim \omega_2$, and ϵ_{21} is the probability of classifying X as in ω_2 when actually $X \sim \omega_1$. If $\lambda_k \geq A$, then X is classified as from ω_1; if $\lambda_k \leq B$, then X is classified as from ω_2; and if $B < \lambda_k < A$, the classifier will take an additional feature measurement, and the process is proceeding to the $(k+1)$th stage. For $m > 2$, the generalized sequential probability ratio test may be used for sequential

[2]For example, a set of orthonormal functions.

[3]Furthermore, in this general treatment, X can be either a (vector-valued) deterministic or a random variable.

classification. If the maximum number of features N available is prespecified the sequential classification procedure must be either truncated at the Nth measurement [21] or a backward computation procedure such as dynamic programming must be used [22]. If all the information required in (3), (6), (9), or (10) is known *a priori*, a pattern classifier can be easily implemented. However, in practice, the quantities in these equations are usually incompletely specified. For example, the w_{ir} in (3) and (6) and the $p(X|\omega_i)$ in (9) and (10) are usually unknown *a priori* or only partially known. Under such circumstances, it is important to introduce a learning process to pattern classifiers such that the unknown information can be estimated (learned) on-line from the actual input pattern samples.

III. Trainable Controllers

The linear classifier shown in Fig. 2 has been used as a trainable controller to realize a switching surface for time-optimal control systems [27], [28]. Using terminologies in pattern classification, the partition of feature space Ω_X is equivalent to the partition of state space, and the switching surface in state space corresponds to the decision surface in feature space. The partitioned regions in state space (feature space) correspond to various control situations (pattern classes). Once the desired switching surface (decision surface) is realized, the controller behaves like a pattern classifier. The output of the time-optimal controller $u = +1$ or -1 represents the classified control situation and also the proper control action in this case. The realization of the switching surface is accomplished through a training procedure.

Since the time-optimal switching surface is, in general, nonlinear, the linear classifier used for the controller is a piecewise linear approximation of the nonlinear switching surface. The state space is first quantized forming elementary hypercubes (elementary control situations) in which control action is assumed constant. Each hypercube is coded with a linearly independent code and it constitutes a pattern (feature) vector; its classification is the same as the control action for the hypercube. A linearly independent code is defined here as one in which the set of pattern vectors representing the zones of a state variable must be a linearly independent set. The dimension of the vectors may be increased by the addition of a $+1$ element to each vector if necessary to produce linear independence.

Two possible linearly independent codes are illustrated in Table I for the state variable x_i. The quantities α, β, and γ are the values of the thresholds which separate the different zones of x_i. The "single-spot" code is so named because the 1 element appears only once in each pattern representation, while the "multispot" code has a multiple number of 1 elements in the pattern representations. Similar codes can be defined with -1, $+1$ elements instead of 0, 1 elements.

The pattern representations (vectors) of the single-spot code are linearly indpendent without the addition of a $+1$ element. The multispot pattern vectors are not linearly independent until they have been augmented with a $+1$ element as shown in Table I. It can be proved [27] that when the state variables are encoded as described, a single linear classifier as shown in Fig. 5 will approximate to an arbitrary degree of accuracy (by increasing the number of quantum zones) switching surfaces of the form

$$f(x_1, x_2, \cdots, x_k) = 0$$

provided that no cross-product terms are included in the expression.[4]

Learning capability of the trainable controller shown in Fig. 5 is accomplished by the adjustable weights $w_1, w_2, \cdots, w_N, w_{N+1}$. The input is the k-dimensional state vector X which is transformed into the N-dimensional vector $[v_1, v_2, \cdots, v_N]^T$. Let

$$V = [v_1, v_2, \cdots, v_N, +1]^T \tag{11}$$

and

$$W = [w_1, w_2, \cdots, w_N, w_{N+1}]^t \tag{12}$$

The output is

$$u = \begin{cases} +1, & \text{if } f(V) > 0 \\ -1, & \text{if } f(V) < 0 \end{cases} \tag{13}$$

TABLE I
Pattern Representation for x_i

Zone of x_i	Single-Spot Code	Multispot Code	Augmented Multispot Code
$x_i > \alpha$	(0,0,0,1)	(1,1,1)	(1,1,1,1)
$\alpha > x_i > \beta$	(0,0,1,0)	(1,1,0)	(1,1,1,0)
$\beta > x_i > \gamma$	(0,1,0,0)	(1,0,0)	(1,1,0,0)
$\gamma > x_i$	(1,0,0,0)	(0,0,0)	(1,0,0,0)

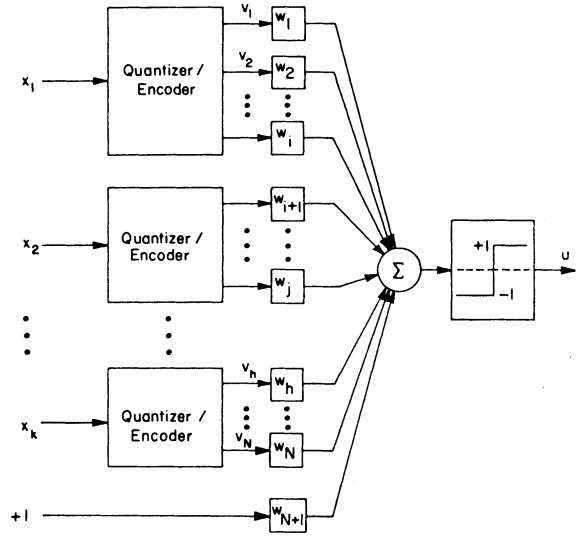

Fig. 5. Trainable controller.

[4]Cross-product terms can be realized by using augmented linear classifiers [27].

where

$$f(V) = V^T W. \tag{14}$$

The switching surface is not known *a priori*, but is defined implicitly by a training set. The training set consists of a finite number of points (control situations) in state whose optimal control actions u^* are known. Specifically, those points in the state space lie on the optimal trajectory $X^*(t)$. Such points, when transformed into the new space Ω_V, define a training set $T = \{V(j), u^*(j)\}, j = 1, \cdots L$. If the set T is decomposed into two sets T_1 and T_2 where all the elements $V(j)$ with $u^* = +1$ are in T_1, and with $u^* = -1$ in T_2, then

$$V^T W > 0, \quad \text{for each } V \in T_1$$
$$V^T W < 0, \quad \text{for each } V \in T_2. \tag{15}$$

The training set T, which is considered as representative of the population of control situations actually encountered, is used to determine a vector W which will then be used to classify other control situations.

During the training process, the trainable controller (Fig. 5) makes changes in its weights based only on the training pattern vector presently being "shown" to it, together with the desired output of that pattern vector. The training pattern vectors are presented to the controller sequentially several times until all pattern vectors (representing control situations) in the training set are being correctly classified, or until the number of classification errors has reached some steady-state value. The weight change after each incorrect classification is αV. Two types of training algorithms, least mean-square error and error correction, may be applied. They are summarized below:

1) *Least Mean-Square Error Training Procedure:* The value of α is

$$\alpha = |\beta\epsilon|/V^T V \tag{16}$$

where $\epsilon = (d - V^T W)$ is defined as the analog error, d is the desired output, and β is a proportionality constant. When the procedure is used and β is small ($\beta << 1$), the controller tends to minimize the mean-square error

$$\bar{\epsilon}^2 = (1/L) \sum_{j=1}^{L} [d(j) - V^T(j) \cdot W]^2$$

where $V(j)$ represents the jth training pattern vector and $d(j)$ the desired output for $V(j)$. The least mean-square error training procedure will give a unique solution weight vector. However, it will not necessarily minimize the number of classification errors even with linearly separable sets T_1 and T_2, i.e., with T_1 and T_2 which can be correctly classified by means of a linear decision surface.

2) *Error-Correction Training Procedure:* In this case, the weight vector is modified when the binary output of the controller disagrees with the desired binary output. That is, if the output is erroneous (i.e., $V^T W < 0$) or undefined (i.e., $V^T W = 0$) for any $V \in T_1$, $V^T W > 0$, then let the new weight vector be

$$W' = W + \alpha V. \tag{17}$$

On the other hand, for $V \in T_2$, if $V^T W \geq 0$, let

$$W' = W - \alpha V. \tag{18}$$

Before training, W may be preset to any convenient values. Three rules for choosing α are suggested [14].

a) Fixed increment rule: α is any fixed positive number.

b) Absolute correction rule: α = the smallest integer greater than

$$|V^T W|/V^T V. \tag{19}$$

c) Fractional correction rule:

$$\alpha = \lambda(|V^T W|/V^T V), \quad 0 < \lambda \leq 2. \tag{20}$$

The error-correction training procedure will find a solution weight vector when T_1 and T_2 are linearly separable. It will not necessarily minimize the number of binary classification errors when T_1 and T_2 are not linearly separable, although it generally does produce close to the minimum number of classification errors.

The idea of using linear classifiers as trainable controllers can be easily extended to a more general synthesis of switching surfaces by means of training techniques [31]. Referring to (8), let $f(X) = 0$ represent the switching surface under study. Initially, the coefficients c_l are unknown, but they certianly can be estimated (learned) through a training procedure. The training sequence or training samples $X(1), X(2), \cdots, X(n), \cdots$ are assumed statistically independent and distributed according to an unknown probability density function $p(X)$. A function of two variables, called the *potential function*, is introduced as

$$K(X, Y) = \sum_{l=1}^{N} \lambda_l^2 \varphi_l(X)\, \varphi_l(Y) \tag{21}$$

where λ_l are real numbers chosen in such a way that the function $K(X, Y)$ is bounded. After n learning samples $X(1), \cdots, X(n)$ are taken, the nth estimate of $f(X)$ is designated by

$$f_n(X) = \sum_{l=1}^{N} c_l(n)\, \varphi_l(X) \tag{22}$$

where

$$c_l(n) = c_l(n-1) + r_n \lambda_l \varphi_l[X(n)]. \tag{23}$$

Similar to the cases for linear classifiers, two types of training algorithms may be applied.[5]

1) *Mean-Square Error Convergence Algorithm:* Let the information of $f(X)$ at $X(1), \cdots, X(n), \cdots$ be measurable but noisy; that is, the actual measurement $y(n)$ can be expressed as

$$y(n) = f(X(n)) + \xi(n) \tag{24}$$

where the $\xi(n)$ are independent random variables (noise with zero mean and finite covariances. Also, the condi-

[5]The method has been called the potential function method [23]–[26]. Only two special algorithms of the method are presented here briefly.

tional probability density function $p(\xi(n)|X(n))$ is not assumed to be a function of n. Then, in the algorithm (23),

$$r_n = \gamma_n[y(n) - f_{n-1}(X(n))] \tag{25}$$

where γ_n is a sequence of positive numbers satisfying

$$\sum_{n=1}^{\infty} \gamma_n = \infty \text{ and } \sum_{n=1}^{\infty} \gamma_n^2 < \infty. \tag{26}$$

It can be shown that

$$\lim_{n \to \infty} E\{[f_n(X) - f(X)]^2\} = 0 \tag{27}$$

that is, the estimated decision function $f_n(X)$ converges to the true decision function $f(X)$ in the mean-square sense.

2) *Error-correction algorithm:* If the information of $f(X)$ is in the form of binary output of a bang-bang controller, for example, let all the training samples with $u^* = +1$ be in training set T_1 and those with $u^* = -1$ be in T_2. The function $f(X)$ used by the controller is such that if

$$\operatorname{sgn} f(X) = +1, \quad \text{then } X \in T_1.$$

and if

$$\operatorname{sgn} f(X) = -1, \quad \text{then } X \in T_2. \tag{28}$$

In this case, the algorithm (23) can be applied with

$$r_n = \tfrac{1}{2}[\operatorname{sgn} f(X(n)) - \operatorname{sgn} f_{n-1}(X(n))]. \tag{29}$$

It can be proved that, in this case,

$$\lim_{n \to \infty} \int_{\Omega_X} |\operatorname{sgn} f_n(X) - \operatorname{sgn} f(X)| p(X)\, dX = 0. \tag{30}$$

IV. Reinforcement Learning Control Systems

Psychologists consider that any systematic change in a system's performance with a certain specified goal is learning. Various kinds of response must be distinguished first in order to describe the performance change of a system. In general, mutually exclusive and exhaustive classes of response $\omega_1, \cdots, \omega_m$ are considered. Let P_i be the probability of occurrence of the ith class of responses. We consider the performance change being expressed by the change or reinforcement of the set of response probabilities $\{P_i\}$. Mathematically, the reinforcement of $\{P_i\}$ can be described as the following relationship [32], [33]:

$$P_i(n+1) = \alpha P_i(n) + (1 - \alpha)\lambda_i(n), \quad n = 0, 1, 2, \cdots \tag{31}$$

where $P_i(n)$ is the probability of the occurrence of ω_i at instant n when the input X is observed

$$0 < \alpha < 1, \quad 0 \le \lambda_i(n) \le 1,$$

and

$$\sum_{i=1}^{m} \lambda_i(n) = 1.$$

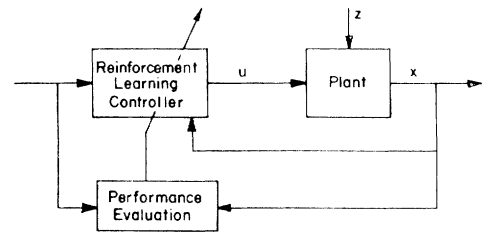

Fig. 6. Reinforcement learning control system.

Because of the relationship between $P_i(n+1)$ and $P_i(n)$ being linear, (31) is often called a linear reinforcement learning algorithm. It can easily be shown that if $\lambda_i(n) = \lambda_i$, then

$$P_i(n) = \alpha^n P_i(0) + (1 - \alpha^n)\lambda_i \tag{32}$$

and

$$\lim_{n \to \infty} P_i(n) = \lambda_i. \tag{33}$$

It is noted that from (33), λ_i is the limiting probability of $P_i(n)$. Hence, $\lambda_i(n)$ should be, in general, related to the information or performance evaluated from the input X at instant n. In learning control systems, the input X to the learning controller is usually the output of the plant and ω_i may directly represent the ith control action. $\lambda_i(n)$ can be identified as the normalized index of performance associated with the ith class of responses (control actions) of the controller. In some simpler cases, $\lambda_i(n)$ may be 0 or 1 to indicate whether the performance of the system at instant n, due to the ith control action, is satisfactory or unsatisfactory. Or $\lambda_i(n)$ may be 0 or 1 to indicate whether or not the decision (or classification) ω_i made by the controller at instant n for the input X is correct. In general, it can be proved that $P_i(n)$ will converge to its maximum as $n \to \infty$ in the mean and in probability if the ith control action is a desired one [35].

The linear reinforcement learning algorithm has been applied to control systems design [33]–[36]. In the design of a reinforcement learning controller, the possible classes of response ω_i, $i = 1, \cdots, m$ of the controller are the corresponding admissible control actions, and the quality of the control actions for different control situations or the performance of the controller is evaluated at the output of the plant. The controller is designed to learn the best control action at each time instant in the absence of complete information about the plant and the environmental disturbance. The learning process is directed by the system's performance evaluated at each time instant.[6] Therefore, the controller is able to learn without an external supervision, or say, to learn "on-line." A block diagram of on-line learning control systems using reinforcement algorithms is shown in Fig. 6.

Waltz and Fu [33] have simulated a class of reinforcement learning control systems on a hybrid computer fa-

[6]The performance-feedback approach used here can certainly be applied also to the controller configurations discussed in Section III. Instead of training, a nonsupervised learning can be achieved with an appropriate choice of performance evaluation.

cility (GEDA-IBM 1620). The feature vector X is essentially the same as the state vector of the plant in this case. The performance index (IP) of the system is of the form

$$\text{IP} = \sum_{n=1}^{N} n[x_1(n)]^2, \quad x_i(n) = x_i \text{ at instant } n\tau \tag{34}$$

where τ is the sampling period which must be long enough to allow for a significant change in X for a typical control action u. The set of admissible control actions $\{u^1, u^2, \cdots, u^m\}$ is given. The controller first classifies any input X into a class of control situations and then learns the best control action for each class of control situations through a linear reinforcement algorithm. The performance evaluated at each time instant n, sometimes called instantaneous performance evaluation or subgoal (IPS), is chosen as

$$\text{IPS}(n) = X^T(n)\, GX(n) \tag{35}$$

where G is a diagonal matrix whose elements may be either preassigned or determined through a learning process.

The classification of control situations in the state space (also the feature space in this case) is performed by constructing adaptive sample sets. As soon as a measurement of X is taken, compare the presently measured vector X with the existing vectors, which have been taken. If the Euclidean distance between X and any existing vector is less than a prespecified distance D, they belong to the same control situation. Otherwise, it is considered as a new control situation, and a new sample set is established with the vector X as its center and D as the radius. If a measured X falls within distance D of two or more existing vectors, it is considered a member of the closest set. The sample set construction produces what might be called a type of generalization since it makes use of the fact that points in the neighborhood of a given point in the state space will usually have similar characteristics and will require similar control actions. The distance D can be varied during the process. The sample sets (control situations) established in the state space must be partitioned into m classes such that a best control action can be determined for each class of control situations. This is accomplished by applying the linear reinforcement learning algorithms.

Let $P_{i,j}(n)$ be the probability that u^i is the best control action for the control situation S_j (or the jth sample set) at instant $n\tau$. Initially, assuming no *a priori* knowledge, all $P_{i,j}(0) = (1/m) \cdot P_{i,j}$ will then be modified according to the following reinforcement algorithm:

$$P_{i,j}(n+1) = \alpha P_{i,j}(n) + (1-\alpha)\,\lambda_{i,j}(n), \qquad 0 < \alpha < 1 \tag{36}$$

where $\lambda_{i,j}(n)$ assumes either 1 or 0 depending upon whether or not the IPS(n) defined in (35) is reduced by applying u^i. α is called the learning parameter. The larger α is, the slower the probabilities $P_{i,j}$ converge, and this results in a slower learning rate. In the process of learning, α can be adjusted according to the amount of reduction in IPS due to the control action u^i. As the learning process proceeds, $P_{i,j}$ approaches 1 for an u^i and each S_j with the possible exception of those sample sets (control situations) located on the decision surfaces (or switching boundaries). A control action u^i is used for control situation S_j with probability $P_{i,j}$ (a pure random strategy) unless some $P_{i,j}$ exceeds a preset threshold. In this case, the u^i for which $P_{i,j}$ is maximum is used as the control action for S_j.

As learning progresses, most of the probabilities $P_{i,j}$ will approach either 1 or 0. If a sample set happens to be located on a decision surface, then some of the probabilities corresponding to this set will oscillate between 1 and 0 during the learning process, since one control action would be the best for one part of the set and a different control action would be the best for another part. It is proposed that these sets should be partitioned into subsets with smaller radii to obtain finer quantization. The procedure is to establish subsets in those sample sets if, after a certain number of X measurements within a sample set S_j, $P_{i,j}$ still lies between two thresholds (typical values of the two thresholds might be 0.1 and 0.9). A typical example of the sample set construction for a second-order plant with two control actions ($m = 2$), $u^1 = +1$ and $u^2 = -1$, is shown in Fig. 7. A sampling period $\tau = 0.5$ second was used. A typical learning curve for the system is shown in Fig. 8. Reasonable performance can be obtained for most stationary systems by applying this subset partition criterion. A second scheme, which can be used for both stationary and nonstationary systems, utilizes the curvature of the approximated (learned) switching boundary to determine where subsets should be established. The chain encoding scheme described by Freeman [39] is used to determine the curvature of the learned switching boundary. Regions of the switching boundary with relatively high curvature in one direction are identified, and those sets that are located on the inside of the curve are further divided into subsets. The utilization of *a priori* knowledge for more efficient partition of state space and subgoal selection has recently been studied [37], [38].

V. Bayesian Learning in Control Systems

In the statistical design of an optimal controller using dynamic programming [42] or statistical decision theory [43]–[45], the true knowledge of the probability distribution of the plant output or of the environmental parameters is required. For example, consider a discrete stochastic plant characterized by the equation

$$X(n+1) = g[X(n), u(n)] \tag{37}$$

where $X(n)$ is the state vector (a random variable) at instant n, and $u(n)$ is the control action at instant n. In determining the optimal control action u^* to minimize the performance index

$$I(n) = E\left\{\sum_{n=1}^{N} F[X(n), u(n-1)]\right\}$$

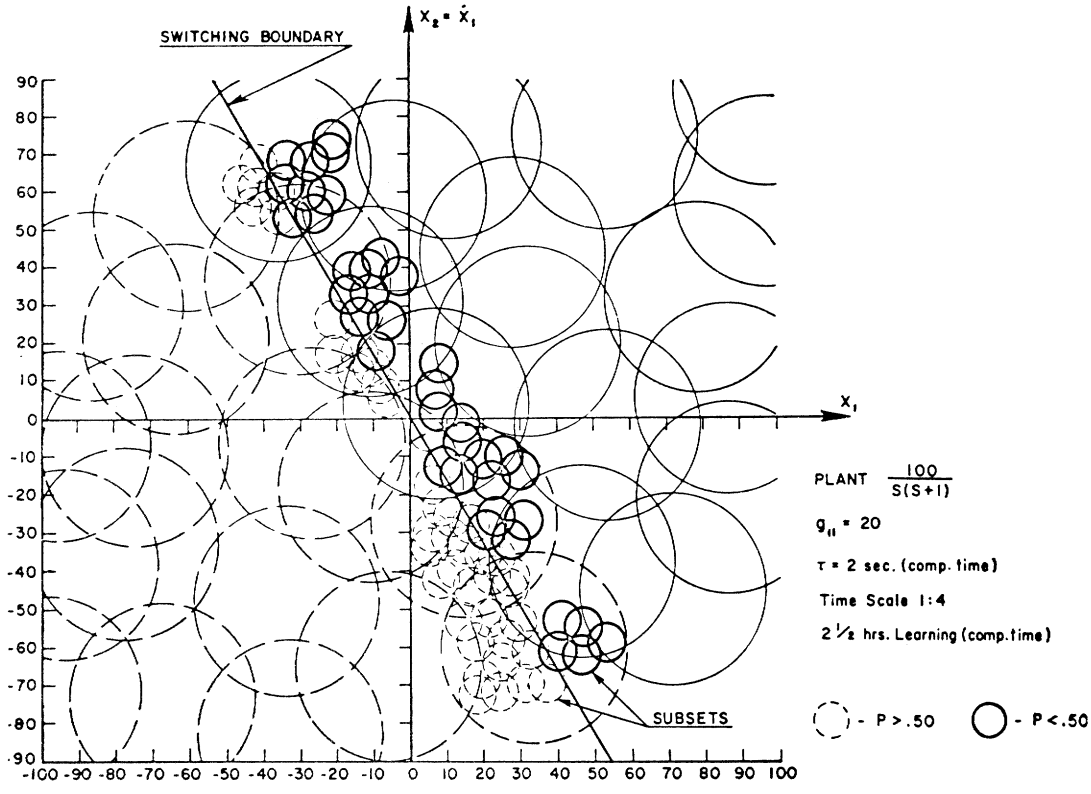

Fig. 7. Typical example of sample set construction.

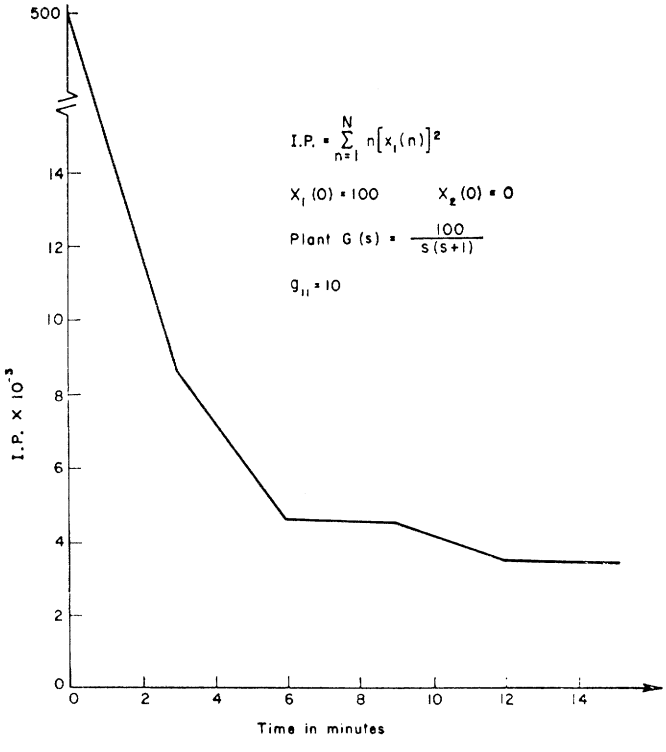

Fig. 8. Typical learning curve.

a recurrence relationship can be derived using dynamic programming with the probability density function $p(X)$ known [42]. Similar to the case mentioned in statistical pattern classification, if the probability distribution or density functions are unknown or incompletely known, a controller can be designed to first estimate (to learn) the unknown function and then to implement the control law on the basis of the estimated information [46]. If the estimated (learned) function approaches the true function, the control law will approach the optimal control law as if all the information required had been known. An approach based on the iterative application of Bayes' theo-

rem to estimate the unknown information is introduced in this section [46]–[49].

Suppose that the probability density function $p(X|\omega_i)$ is to be learned, where ω_i represents the ith class of control situations. Let $X(1), \cdots, X(n)$ be the feature measurements with known classifications of control situations (called learning samples), say, all in ω_i. This is certainly the case of supervised learning. If the form of $p(X|\omega_i)$ is known but some parameters θ are unknown, then the problem is reduced to that of estimating θ for given measurements $X(1), \cdots, X(n)$. Since θ is unknown, it can be assumed to be a random variable with a certain *a priori* distribution. By applying Bayes' theorem, the *a posteriori* density function of θ is computed from the *a priori* density function and the information obtained from sample measurements; i.e.,

$$p[\theta|\omega_i, X(1), \cdots, X(n)] = \frac{p[X(n)|\omega_i, \theta, X(1), \cdots, X(n-1)]\, p[\theta|\omega_i, X(1), \cdots, X(n-1)]}{p[X(n)|\omega_i, X(1), \cdots, X(n-1)]}. \tag{38}$$

For example, if $p(X|\omega_i)$ is Gaussian distributed with mean vector M and covariance matrix K, and the unknown parameter θ is the mean vector M, let the *a priori* distribution of θ, $p_0(\theta|\omega_i)$ be also Gaussian distributed with initial mean vector $M(0)$ and initial covariance matrix $\Phi(0)$. Then, after the first sample measurement $X(1)$ has been taken

$$p[\theta|\omega_i, X(1)] = \frac{p[X(1)|\omega_i, \theta]\, p_0(\theta|\omega_i)}{p[X(1)|\omega_i]}. \tag{39}$$

It is noted that the assumption of a Gaussian distribution for $p_0(\theta|\omega_i)$ will simplify the computation of (39) since the product of $p[X(1)|\omega_i, \theta]\, p_0(\theta|\omega_i)$ is also a Gaussian distribution. By using this property of reproducing distribution of $p_0(\theta|\omega_i)$ and the iterative application of Bayes' theorem, after n learning samples, a recursive expression for estimation $\theta = M$ is given as [47], [48]

$$M(n) = K[\Phi(n-1) + K]^{-1} M(n-1) + \Phi(n-1)[\Phi(n-1) + K]^{-1} X(n) \tag{40}$$

$$\Phi(n) = K[\Phi(n-1) + K]^{-1} \Phi(n-1). \tag{41}$$

In terms of the initial estimates $M(0)$ and $\Phi(0)$, (40) and (41) become

$$M(n) = n^{-1}K[\Phi(0) + n^{-1}K]^{-1} M(0) + \Phi(0)[\Phi(0) + n^{-1}K]^{-1} \langle X \rangle \tag{42}$$

$$\Phi(n) = n^{-1}K[\Phi(0) + n^{-1}K]^{-1} \Phi(0) \tag{43}$$

where $\langle X \rangle = (1/n) \sum_{i=1}^{n} X(i)$ is the sample mean. Equation (42) shows that the nth estimate of the mean vector $M(n)$ can be interpreted as a weighted average of the *a priori* mean vector $M(0)$ and the sample information $\langle X \rangle$. As $n \to \infty$, $M(n) \to \langle X \rangle$ and $\Phi(n) \to 0$ which means, on the average, the estimate $M(n)$ will approach the true mean vector M. Similarly, if the covariance matrix K is unknown or if both M and K are unknown, the Bayesian learning technique can also be applied [49].

If the correct classifications of the learning samples $X(1), \cdots, X(n)$ are not available, a nonsupervised learning technique must be used. In this case, each measurement $X(i)$ may be considered to come from any one of the m classes of control situations. A relatively general approach is to form a mixture density (or distribution) function on the basis of the probability density functions from all possible classifications, i.e.,

$$p(X|\theta, P) = \sum_{i=1}^{m} P_i p(X|\omega_i, \theta_i) \tag{44}$$

where θ_i is the unknown parameter associated with $p(X|\omega_i)$ and $\theta = \{\theta_i;\ i = 1, \cdots, m\}$, $P = \{P_i;\ i = 1, \cdots, m\}$. Let $B = (\theta, P)$ and consider that the sequence of independent measurements $X(1), \cdots, X(n)$ are taken from the mixture with probability density function $p(X)$. Then, a successive application of Bayes' theorem gives

$$p[B|X(1), \cdots, X(n)] = \frac{p[X(n)|X(1), \cdots, X(n-1), B]\, p[B|X(1), \cdots, X(n-1)]}{p[X(n)|X(1), \cdots, X(n-1)]} \tag{45}$$

It is necessary to select the *a priori* probability $p_0(B)$ which is not equal to zero at the true value of B characterizing the mixture under consideration. Also, the identifiability conditions for a given type of mixture must be imposed in order to uniquely learn the unknown parameters. Whether the mixture $p(X|\theta, P)$ is identifiable or not is a problem of unique characterization. That is, for a particular family of the ith components (parameters conditional) density functions $\{p(X|\omega_i, \theta_i)\}$ and a set of parameters θ and P, the mixture $p(X|\theta, P)$ uniquely determines the sets of parameters $\{\theta_i\}$ and $\{P_i\}$. It is then clear that if the nonsupervised learning problem is such that the mixture is not uniquely characterized by $\{\theta_i\}$ and $\{P_i\}$ (not identifiable), then there exists no unique solution to the underlying estimation problem. In addition to the Bayesian learning technique [51], [52], the stochastic approximation procedure discussed in Section VI can also be applied for estimating unknown parameters involved in a mixture distribution [53].

VI. Learning Control Systems Using Stochastic Approximation

The learning control systems discussed in Sections IV and V have demonstrated the advantages of introducing learning into a control system when the *a priori* information required is incompletely known. A more general design technique using the performance feedback approach is discussed in this section. The basic idea is the application of the stochastic approximation procedure to

the design of a learning controller [55]–[58]. In other words, the controller uses the stochastic approximation procedure to learn the best control action for each class of control situations. In order to implement the idea, the following approach is taken. First, a proper evaluation of system performance must be performed such that the performance evaluation can be used to direct the learning process. However, since in learning control problems the plant-environment characteristics are, in general, unknown or incompletely known, an exact evaluation of the performance index is actually impossible. In addition, an instantaneous (or an interval basis) performance evaluation (a subgoal) must be appropriately chosen such that the system's learning directed by the instantaneous performance evaluation will guarantee the final optimality with respect to the overall performance index specified. Under such a circumstance, it is proposed that the stochastic approximation procedure be applied to estimate the performance index first and then to learn the best control action.

Consider a plant described by the equation

$$y(n+1) = \Phi[y(n), u(n+1)] \tag{46}$$

where $y(n+1)$ is the observed response of the plant at instant $n+1$ when the control action $u(n+1)$ is applied. The instantaneous performance evaluation is chosen as

$$z(n+1) = f[y(n+1), u(n+1), y(n)] \tag{47}$$

where f is a prespecified positive definite function. For a stationary stochastic plant, the conditional density function $p[z(n+1)|u(n), y(n), u(n+1)]$ does not depend explicitly on n; i.e.,

$$p[z(n+1)|u(n) = u^r, y(n) = y, u(n+1) = u^j] = p[z|u^r, y, u^j] \tag{48}$$

for every n. The performance index of the system is

$$\text{IP} = E[z|u^r, y, u^j]. \tag{49}$$

The optimal control action u^* is defined by

$$E[z|u^r, y, u^*] = \min_{j=1,\cdots m} \{E[z|u^r, y, u^j]\}. \tag{50}$$

Since $p[z|u^r, y, u^j]$, $j = 1, \cdots, m$, and Φ are unknown, $E[z|u^r, y, u^j]$ can only be obtained from the successive estimates $\hat{E}_{N_j}[z|u^r, y, u^j]$, $N_j = 1, 2, \cdots$, which converge to $E[z|u^r, y, u^j]$ with probability one for every u^j. Also, since the condition associated with the estimation of $E[z|u^r, y, u^j]$ is always (u^r, y, u^j), let (u^r, y, u^j) be (X^q, u^j). Then

$$E[z|u^r, y, u^j] = E[z|X^q, u^j]. \tag{51}$$

Let $z_{N_{qj}+1}$ designate the value of $z(n+1)$ distributed according to $p(z|X^q, u^j)$ where N_{qj} is the number of times in n instants that u^j followed the occurrence of X^q. The stochastic approximation procedure is used to estimate $E(z|X^q, u^j)$; i.e.,

$$\hat{E}_{N_{qj}+1}[z|X^q, u^j] = \hat{E}_{N_{qj}}[z|X^q, u^j] + \gamma_{N_{qj}}\{z(N_{qj}+1) - \hat{E}_{N_{qj}}[z|X^q, u^j]\} \tag{52}$$

for $N_{qj} = 0, 1, 2, \cdots$, where $\hat{E}_0[z|X^q, u^j] = 0$ and $\gamma_{N_{qj}} = 1/N_{qj}$. A block diagram to illustrate the operation of (52) is shown in Fig. 9.[7] Then

$$P\{\lim_{N_{qj}\to\infty} \hat{E}_{N_{qj}}[z|X^q, u^j] = E[z|X^q, u^j]\} = 1. \tag{53}$$

The controller is designed to use a pure random strategy to choose the proper control action at each instant. The desired optimal control law is

$$P(u^*|X^q) = P_{u^*,q} = 1. \tag{54}$$

The subjective probabilities $\{P_{k,q}(n_q); u^k, k = 1, \cdots, m$, following the occurrence of $X^q\}$ for the pure random strategy are modified on the basis of the estimates $\hat{E}[z|X^q, u^j]$. n_q is the number of occurrences of X^q in n instants and

$$n_q = \sum_{j=1}^{m} N_{qj}.$$

Several algorithms can be applied to modify the subjective probabilities [56]. The algorithm described in the following is the one based on the stochastic approximation procedure. After $(n_q + 1)$ occurrences of X^q, let the estimates of the performance indices be $\hat{E}_{n_q+1}[z|X^q, u^j]$, $j = 1, \cdots, m$. The subjective probabilities are recursively computed for every u^k, $k = 1, \cdots, m$, by

$$P_{k,q}(n_q+1) = P_{k,q}(n_q) + \gamma_{n_q+1} \cdot [\xi_{k,q}(n_q+1) - P_{k,q}(n_q)] \tag{55}$$

where

$$(1-\gamma_{n_q}) > 0, \quad \sum_{n_q=1}^{\infty} \gamma_{n_q}^2 < \infty, \quad \prod_{n_q=1}^{\infty} (1-\gamma_{n_q}) = 0$$

and

$$\sum_{n_q=r}^{\infty} \prod_{k=r}^{n_q} (1-\gamma_k)^2 < \infty, \quad \text{for} \quad r = 0, 1, 2, \cdots$$

and

$$\xi_{k,q}(n_q+1) = \begin{cases} 1, & \text{if } \hat{E}_{n_q+1}[z|X^q, u^k] = \min\limits_j \hat{E}_{n_q+1}[z|X^q, u^j] \\ 0, & \text{if } \hat{E}_{n_q+1}[z|X^q, u^k] \neq \min\limits_j \hat{E}_{n_q+1}[z|X^q, u_j]. \end{cases} \tag{56}$$

It can be shown that if, for every suboptimal control action u^μ [55],

$$\sum_{n_q=1}^{\infty} \gamma_{n_q} E[\xi_{\mu,q}(n_q)|z(1), \cdots, z(n_q)] < \infty \tag{57}$$

[7]For an example of the physical realization of stochastic approximation procedure see [101].

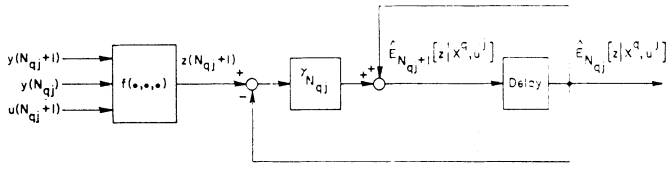

Fig. 9. Algorithm (52).

then

$$P\{\lim_{n_q \to \infty} P_{u^*,q}(n_q) = 1\} = 1. \tag{58}$$

Equation (58) indicates that the desired optimal control law as defined in (54) will eventually be obtained with probability one. If the plant response is continuously distributed, it has been proposed in [55], [7] that the potential function method can be applied to estimate the performance index.

In the case where the optimal control law is of the form [58]

$$\begin{aligned} u(n) &= h[y(n-1), y(n-2), \cdots, y(n-k)] \\ &= h[Y(n)] \end{aligned} \tag{59}$$

let the control law be represented by

$$h[Y(n)] = \sum_{l=1}^{N} c_l \varphi_l[Y(n)] \tag{60}$$

where $\varphi_l(Y)$, $l = 1, \cdots, N$, are linearly independent functions. The coefficients c_l can be estimated by applying the stochastic approximation procedure, and the nth estimate of $h(Y)$ is designated as

$$\hat{u}(n) = \hat{h}[Y(n)] = \sum_{l=1}^{N} c_l(n)\, \varphi_l[Y(n)]. \tag{61}$$

Hence, the estimated control $\hat{u}(n)$, will approach $u(n)$ in the mean-square sense.

VII. Stochastic Automata as Models of Learning Controllers

The reinforcement learning control system has been recently formulated mathematically by way of stochastic automata theory [74], [75]. A stochastic automaton is a quintuple (Y, Q, U, F, G) [63]–[66] where Y is a finite set of inputs, $Y = \{y^1, \cdots, y^r\}$, Q is a finite set of states, $Q = \{q^1, \cdots, q^s\}$, U is a finite set of outputs, $U = (u^1, \cdots, u^m\}$, F is the next state function

$$q(n+1) = F[y(n), q(n)], \tag{62}$$

and G is the output function

$$u(n) = G[q(n)]. \tag{63}$$

In general, the function F is stochastic and the function G may be deterministic or stochastic.

For each input y^k applied to the automaton at time instant n, the function F is usually represented as a state transition probability matrix $M^k(n)$. The i, j element $p_{ij}^k(n)$ of $M^k(n)$ is defined by

$$p_{ij}^k(n) = P\{q(n+1) = q^j | q(n) = q^i,\ y(n) = y^k\} \tag{64}$$

$$\sum_{j=1}^{s} p_{ij}^k(n) = 1. \tag{65}$$

Thus, the probability distributions of state $q(n+1)$ is determined by the probability distribution of $q(n)$ and $y(n)$. It is easily seen that a deterministic finite automaton is a special case of stochastic automaton; the matrix M^k consists of zeros and ones, and each row of the matrix contains exactly one element which is equal to unity. The function G, if it is stochastic, can also be represented in a matrix form. The i, j element of the matrix is the probability of the output being u^j if the state is q^i. However, in this application, the output function G is considered as a deterministic function of the state only. If the state transition probability matrices are $M(1)$, $M(2)$, $\cdots$, for inputs $y(1)$, $y(2)$, $\cdots$, respectively, the state transition probability matrix for a sequence of inputs can be found by simple matrix multiplication.

Because of the stochastic nature in state transitions, stochastic automata are considered suitable for modeling learning systems [67], [75]. The algorithms which modify the system's structure or parameter values should provide certain properties to guarantee the necessary improvement of the system's performance. One approach suggested is the modification of state transition probabilities of a stochastic automaton such that its performance can be improved during operation. Another approach proposed is the modification of state probabilities; i.e., the probabilities for the automaton at each state. In both cases, the stochastic automaton has a variable structure due to the change of transition probabilities or state probabilities. Consequently, the feature of variable structure results in the learning behavior of the automaton. Reinforcement algorithms have been suggested for modifying state transition probabilities or state probabilities. The amount of reinforcement is, in general, a function of the automaton's performance. The new distribution or probabilities reflect the information which the automaton has received from the input and consequently its ability to improve its performance. Varshavsky and Vorontsova [70], following Tsetlin's approach for deterministic automata [71], have used the variable structure stochastic automata as models for learning systems operating in random environments (Fig. 10). At each step, the performance of the (learning) automaton through the environment is evaluated by either a penalty (or unsatisfactory performance) ($y = 1$), or a nonpenalty (or satisfactory performance ($y = 0$). If the output of the automaton is u^j, $j = 1, 2, \cdots, m$, the random environment generates a penalty with probability π^j or a nonpenalty with probability $(1 - \pi^j)$. The overall measure of the performance of an automaton is the mathematical expectation of penalty

$$I = \lim_{n \to \infty} (1/n) \sum_{j=1}^{n} y(j). \tag{66}$$

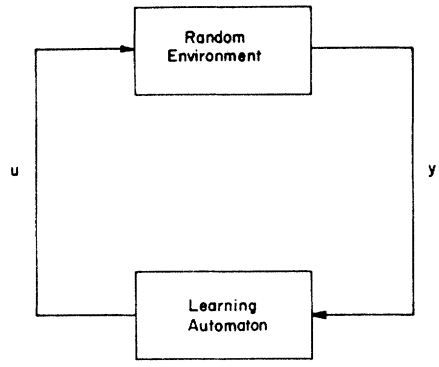

Fig. 10. Automaton operating in random environment.

If

$$I < (1/m) \sum_{j=1}^{m} \pi^j, \tag{67}$$

the performance of the automaton is called expedient, expediency being defined as closeness of I to $I_{\min} = \min(\pi^1, \cdots, \pi^m)$. In the ideal case, $I = I_{\min}$ and the automaton is said to have optimal performance. In order to achieve expedient performance, modifications of p_{ij}^k are proposed. The idea is to decrease p_{ij}^k if the transition from q^i to q^j due to y^k is followed by a penalty. On the other hand, if a nonpenalty follows, p_{ij}^k is increased. The remaining elements in the state transition probability matrix p_{ih}^k, $h \neq j$, are also varied so to keep condition (65) satisfied.

The mathematical formulations of reinforcement algorithms for modifying transition probabilities or state probabilities can be classified as 1) the linear reinforcement algorithm, and 2) the nonlinear reinforcement algorithm. Only the linear reinforcement algorithm will be briefly discussed. Very little results have been obtained so far concerning the nonlinear reinforcement algorithm [67], [73]. Although the basic formulations presented here are based on the modification of state probabilities, similar formulations can be easily derived for the modification of transition probabilities.

Let the state probability of a stochastic automaton in state q^i at the time n be $p_i(n)$; i.e.,

$$p_i(n) = P\{q(n) = q^i\}.$$

For the linear reinforcement algorithm, similar to (31), $p_i(n+1)$ and $p_i(n)$, for all i, are related by a linear relationship

$$p_i(n+1) = \alpha p_i(n) + (1-\alpha)\,\lambda_i(n), \quad i = 1, \cdots, s \tag{68}$$

where $0 < \alpha < 1$ and $0 \leq \lambda_i(n) \leq 1$, $\Sigma_{i=1}^{s} \lambda_i(n) = 1$. It can be easily shown that, if $\lambda_i(n) = \lambda_i$, then

$$\lim_{n \to \infty} p_i(n) = \lambda_i, \tag{69}$$

i.e., λ_i is the limiting probability of $p_i(n)$. Hence, $\lambda_i(n)$ should be related to the information or performance measure evaluated from the response through the stochastic environment due to the ith action (output) of the (learning) automaton. To illustrate the reinforcement algorithm for changing the structure of the automaton and the subsequent properties, assume here that the learning automaton will search for the u to maximize the expected value of λ.[8] If $u(n) = u^i$ or it is equivalent to say that y is from ω_i, $y \sim \lambda\ \omega_i$, then

$$p_i(n+1) = \alpha p(n) + (1-\alpha)\,\lambda_i(n)$$

and

$$p_j(n+1) = \alpha p_j(n) + (1-\alpha)[1/(m-1)] \cdot [1 - \lambda_i(n)], \quad j \neq i. \tag{70}$$

Let

$$E_\lambda\{\lambda_i(n)\} = \int \lambda_p(\lambda | y \sim \omega_i)\, d\lambda = m_i \tag{71}$$

and

$$\gamma_i = \prod_{\substack{j=1 \\ \neq i}}^{m} (1 - m_j) \Big/ \sum_{i=1}^{m} \prod_{\substack{j=1 \\ \neq i}}^{m} (1 - m_j). \tag{72}$$

It can be shown [67], [74] that

$$\lim_{n \to \infty} E_\lambda\{p_i(n)\} = \gamma_i, \quad i = 1, \cdots, m \tag{73}$$

and $E_\lambda\{p_i(n)\}$ will converge to γ_i monotonically with n. Since the m_i are in the same magnitude order as the corresponding γ_i in the limit as $n \to \infty$, the larger (better) that m_i is, the higher the limit γ_i will be. In addition, the largest (best) m_i corresponds to the highest γ_i, consequently to the highest expected probability.

In learning control problems, the stochastic automaton becomes the learning controller. $y(n)$ is the observed response from the plant environment at time n. The basic problem is to select a control action u^* from the set of admissible control actions $\{u^1, \cdots, u^m\}$ such that

$$E\{\xi | y, u^*\} = \min_k \{E[\xi | y, u^k]\} \tag{74}$$

where $\xi = f(y, u^i, y')$ is an instantaneous performance evaluation of the action u^i following the observation y, and y' is the response due to action u^i applied to the plant after y was observed. It can easily be seen that $E\{\xi | y, u^i\}$ is the performance index (expressed as the expectation of ξ with respect to the probability density function $p(\xi | y, u^i)$) for the action u^i applied after the observation of y. The optimal control law is given by

$$p\{u = u^* | y\} = 1. \tag{75}$$

Let

$$p_i(n) = P\{u^i | y(n)\}, \quad i = 1, \cdots, m. \tag{76}$$

Then the linear reinforcement algorithm corresponding to this case is of the following form:

$$p_i(n+1) = \alpha p_i(n) + (1-\alpha)\,\lambda_i(n+1), \quad i = 1, \cdots, m \tag{77}$$

[8]The reinforcement algorithm for minimizing the expected value of λ can be derived analogously.

where

$$\lambda_i(n+1) = \begin{cases} 1, \text{ if } \hat{E}_{ni}\{\xi|y, u^i\} = \min\limits_k \hat{E}_{nk}\{\xi|y, u^k\} \\ 0, \text{ if } \hat{E}_{ni}\{\xi|y, u^i\} \neq \min\limits_k \hat{E}_{nk}\{\xi|y, u^k\}. \end{cases} \tag{78}$$

The control action applied after $(n+1)$th observation of y can be selected from $\{u^1, \cdots, u^m\}$ using a pure random strategy based on $p_i(n)$, $i = 1, \cdots, m$, $n = 1, 2, \cdots$. After $(n+1)$ performance evaluations following the occurrences of y, let

$$p\{\hat{E}_{nj}[\xi|y, u^j] = \min_k [\hat{E}_{nk}(\xi|y, u^k)]\} = z_j(n+1). \tag{79}$$

Then, from (77)–(79),

$$E[p_j(n+1)|p_j(n)] = p_j(n) + (1-\alpha)\, z_j(n+1). \tag{80}$$

If

$$P\{\lim_{n\to\infty} z_j(n) = 0\} = 1, \quad \text{for every } u^j \neq u^* \tag{81}$$

$$p_j(0) > 0, \sum_{j=1}^{m} p_j(0) = 1 \tag{82}$$

and

$$p_i(n) = \max_{u^k} \{p_k(n)\}, \quad i, k = 1, \cdots, m \tag{83}$$

then it is sufficient for (77) and (78) to yield [35]

$$P\{\lim_{n\to\infty} P[u^*|y(n)] = 1\} = 1. \tag{84}$$

That is, the desired optimal control law as defined in (75) will eventually be obtained with probability one.

VIII. Conclusions and Remarks

The basic concept of learning control has been reviewed. Several important learning techniques with direct applications to control system design have been described. Perhaps only time can tell whether or not these approaches will coexist peacefully, or whether some of the approaches will soon be retired, polished, or merged. Theoretically speaking, these learning algorithms have similar learning properties [8], [93]. However, from an engineering viewpoint, the a priori information required and the computation involved for these techniques are different. For example, in applying Bayesian estimation, the computation will usually be quite complicated if a reproducing *a priori* density function for the unknown parameter is not available. Many other techniques in pattern recognition, learning, and estimation may have potential applications to control problems. However, they have not been included in this review in order to keep the paper at a reasonable length. Learning control is still a new area of research.[9] Preliminary attempts to apply theoretical results to spacecraft control problems have already been made by several authors [29], [80], [81], [98]. Other applications include the control of valve actuators [82], and the control of power systems and production processes [83]–[85]. At the present state of the art, the implementation of more sophisticated on-line learning techniques usually requires large or high-speed computers. Nevertheless, with the rapid progress in computer technology, it is anticipated that the seriousness of this problem will be reduced.

In supervised or off-line learning (or training) schemes, the system usually stops to learn as soon as the training process is terminated. When the system is actually operating within its random environment, nonsupervised or on-line learning schemes must be used. It is known that the rate of learning for nonsupervised learning is relatively slower than that for supervised learning. In many practical situations, it is possible to use the combination of both supervised and nonsupervised learning schemes. That is, a supervised learning scheme is used first to learn as much a priori information as possible, and then a nonsupervised learning scheme will be in operation on-line. The operation of such a system can be considered as consisting of two modes: training and on-line learning. In practical design, the training process can usually be performed as a computer simulation or by a man–computer interactive system.

In the theoretical study, the following problems should be interesting to researchers.

1) *Learning in Nonstationary Environment:* Most of the existing learning algorithms are valid only in a stationary environment (estimating stationary parameters). Because of the plant dynamics involved or possibly nonstationary (unknown) environmental disturbance, algorithms for learning in nonstationary environment need to be developed. Preliminary attempts have been made by employing a two-step learning algorithm [62] or nonlinear reinforcement. If the nonstationary environment can be approximated by a finite number of different stationary environments (switching environments), pattern recognition or mixture decomposition techniques may be applied to identify the stationary environment in which the system is operating [88], [89], [99]. Then the corresponding learning algorithm for that environment can be used. Also, nonlinear reinforcement algorithms, because of some mathematical difficulties involved in the analysis, have not been fully explored.

2) *Improvement of Learning Rate:* The rate of learning

[9]Since the area of learning control is new and still immature, it is very difficult to review the subject on the basis of a unified framework at this stage. It may be felt that the approaches described in this paper were drawn primarily from classification (decision) and estimation theory, psychology of learning, and theory of stochastic automata. Also, sometimes, in order to obtain relatively simple but practical solutions, suboptimal (but satisfactory) and heuristic approaches may be preferred [87], [92], [102]. Nevertheless, because of the learning behavior required, it is anticipated that the problems of learning control will continue to have the flavor of both control and artificial intelligence. (The latter is, of course, also a new area of research.)

(or learning time) of existing learning algorithms are considered rather slow, particularly for fast response systems. It may be improved either by appropriate utilization of *a priori* knowledge [20], [38] (for example, the form of the plant equation, some parameter values of their ranges, or the type of the environmental disturbance may be known), or by developing new faster learning algorithms.

3) *Stopping Rule:* Most of the existing learning algorithms are proved to be convergent asymptotically. In other words, the true parameter values will be obtained as the number of learning samples approaches infinity. In practice, with a finite time of operation, the learning samples available may be limited to a rather smaller number. Under such a circumstance, the information learned at finite time becomes important. Hence, in addition to the asymptotic convergence, the property of learning algorithms at a finite number of iterations needs to be investigated. On the other hand, if the tolerable or satisfactory (but usually not the optimal) performance of the system can be prespecified, an appropriate stopping rule is required so that the learning time will not be longer than necessary [100].

4) *Hierarchical Structure of Learning:* In rather complex learning situations, several learning algorithms may be introduced at different levels of information gathering. Usually these learning processes at different levels or in different learning loops are related to each other. The performance of learning at a lower level [e.g., the learning of subjective probabilities in (55)] depends upon the learned information at a higher level [e.g., the learning of IP in (52)].[10] If the learning at a higher level always produces correct information (e.g., in the form of a perfect teacher), the learning at a lower level depends only on the algorithm implemented. If the learning at a higher level is implemented by another learning algorithm, then, in general, it only produces perfectly correct information asymptotically. (It is sometimes called the "unreliable teacher.") In this case, the convergence of the overall system learning requires special attention, even though the convergence of the learning process at each level has been guaranteed. The effects on learning performance in systems using more than one learning algorithm and in those with an unreliable teacher should also be interesting problems for further study [54], [60], [90].

5) *Fuzzy Set Approach to Learning Control:* Recently, the concept of fuzzy set has been applied to the design of learning control systems [94]–[96]. The basic idea of fuzzy set has been motivated by practical engineering problems. It is certainly interesting to see further studies of this approach with respect to the applications in control theory and practice.

Acknowledgment

The author wishes to thank the reviewers for their comments and suggestions.

[10]The learned IP is used to direct the learning process of subjective probabilities.

References

A. General

[1] H. W. Mosteller, "Learning control systems," GE Advanced Electronics Center, Ithaca, N.Y., Tech. Rept. R64 ELC37, March 1964.

[2] K. S. Fu, "Learning control systems," in *Computer and Information Sciences*, J. T. Tou and R. H. Wilcox, Eds. Washington, D.C.: Spartan, 1964.

[3] J. Sklansky, "Learning systems for automatic control," *IEEE Trans. Automatic Control*, vol. AC-11, pp. 6–19, January 1966.

[4] J. T. Tou and J. D. Hill, "Steps toward learning control," *Proc. 1966 JACC* (Seattle, Wash.), pp. 12–26.

[5] J. M. Mendel, "Survey of learning control systems for space vehicle applications," *Proc. 1966 JACC* (Seattle, Wash.), pp. 1–11.

[6] J. E. Gibson, K. S. Fu, J. D. Hill, J. A. Luisi, R. H. Raible, and M. D. Waltz, "Philosophy and state of the art of learning control systems," Purdue University, Lafayette, Ind., Tech. Rept. TR-EE63-7, November 1963.

[7] K. S. Fu, "Learning control systems," in *Advances in Information Systems Science*, J. T. Tou, Ed. New York: Plenum, 1969.

[8] ——, "Learning system theory," in *System Theory*, L. A. Zadeh and E. Polak, Eds. New York: McGraw-Hill, 1969, ch. 11.

[9] C. T. Leondes and J. M. Mendel, "Artificial intelligence control," McDonnel-Douglas Astronautics Co., Santa Monica, Calif., Tech. Rept. 4336, January 1967.

[10] Ya. Z. Tsypkin, *Adaptation and Learning in Automatic Systems* (English transl.). New York: Academic Press (to be published).

[11] Ya. Z. Tsypkin, G. K. Kelmans, and L. E. Epstein, "Learning control systems," *Proc. 1969 IFAC Cong.* (Warsaw, Poland).

[12] R. L. Stratonovich, "Does there exist a theory of synthesis of optimal adaptive, self-learning, and self-adaptive systems?" *Avtomat. i Telemekh.*,vol. 29, pp. 83–92, January 1968.

[13] Ya. Z. Tsypkin, "All the same, does a theory of synthesis of optimal adaptive systems exist?" *Avtomat. i Telemekh.*, vol. 29, pp. 93–98, January1968.

B. Pattern Classification

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

[15] G. Sebestyen, *Decision-Making Processes in Pattern Recognition*. New York: Macmillan, 1962.

[16] L. N. Kanal, Ed., *Pattern Recognition*. Washington, D.C.: Thompson Book Co., 1968.

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

[18] Y. C. Ho and A. K. Agrawala, "On pattern classification algorithms—introduction and survey," *IEEE Trans. Automatic Control* (Expository Papers), vol. AC-13, pp. 676–690, December 1968.

[19] C. K. Chow, "An optimum character recognition system using decision functions," *IRE Trans. Electronic Computers*, vol. EC-6, pp. 247–254, December 1957.

[20] K. S. Fu, *Sequential Methods in Pattern Recognition and Machine Learning*. New York: Academic Press, 1968.

[21] Y. T. Chien and K. S. Fu, "A modified sequential recognition machine using time-varying stopping boundaries," *IEEE Trans. Information Theory*, vol. IT-12, pp. 206–214, April 1966.

[22] K. S. Fu, Y. T. Chien, and G. P. Cardillo, "A dynamic programming approach to sequential pattern recognition," *IEEE Trans. Electronic Computers*, vol. EC-16, pp. 790–803, December 1967.

[23] M. A. Aiserman, E. M. Braverman, and L. I. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition," *Avtomat. i Telemekh.*, vol. 25, pp. 917–936, 1964.

[24] ——, "The method of potential functions for the problem of restoring the characteristic of a function converter from randomly observed points," *Avtomat. i Telemekh.*, vol. 25, pp. 1705–2213, 1964.

[25] ——, "The probability problem of pattern recognition learning and the method of potential functions," *Avtomat. i Telemekh.*, vol. 25, pp. 1307–1323, 1964.

[26] E. M. Braverman and E. S. Pyatnitskii, "Estimation of the rate of convergence of algorithms based on the potential function method," *Avtomat. i Telemekh.*, vol. 27, pp. 95–112, January 1966.

C. Trainable Controller

[27] F. W. Smith, "Contact control by adaptive pattern-recognition techniques," Stanford Electronics Laboratory, Stanford University, Stanford, Calif., Tech. Rept. 6762-2, April 1964.

[28] B. Widrow and F. W. Smith, "Pattern recognizing control systems," in *Computer and Information Sciences*, J. T. Tou and R. H. Wilcox, Eds. Washington, D.C.: Spartan, 1964.
[29] F. B. Smith, Jr., *et al.*, "Trainable flight control system investigation," Wright-Patterson Air Force Base, Dayton, Ohio, Tech. Rept. FDL-TDR-64-89, 1964.
[30] A. R. Butz, "Learning bang-bang regulators," *Proc. 1968 Hawaii Internatl. Conf. Sys. Sci.*
[31] J. M. Mendel and J. J. Zapalac, "The application of techniques of artificial intelligence to control system design," in *Advances in Control Systems*, vol. 6, C. T. Leondes, Ed. New York: Academic Press, 1968.

D. Reinforcement Learning Control Systems

[32] R. Bush and F. Mosteller, *Stochastic Models for Learning*. New York: Wiley, 1955.
[33] M. D. Waltz and K. S. Fu, "A heuristic approach to reinforcement learning control systems," *IEEE Trans. Automatic Control*, vol. AC-10, pp. 390–398, October 1965.
[34] S. J. Kahne and K. S. Fu, "Learning system heuristics," *IEEE Trans. Automatic Control* (Correspondence), vol. AC-11, pp. 611–612, July 1966.
[35] K. S. Fu and Z. J. Nikolic, "On some reinforcement techniques and their relation to the stochastic approximation," *IEEE Trans. Automatic Control* (Correspondence), vol. AC-11, pp. 756–758, October 1966.
[36] J. D. Lambert and M. D. Levine, "Learning control heuristics," *IEEE Trans. Automatic Control* (Correspondence), vol. AC-13, pp. 741–742, December 1968.
[37] L. E. Jones, III, "On the choice of subgoals for learning control systems," *IEEE Trans. Automatic Control*, vol. AC-13, pp. 613–621, December 1968.
[38] L. E. Jones, III and K. S. Fu, "On the selection of subgoals and the use of a priori information in learning control systems," *Automatica*, November 1969.
[39] H. Freeman, "On the digital computer classification of geometric line patterns," *Proc. 1962 NEC*, vol. 18.
[40] J. D. Lambert and M. D. Levine, "A two-stage learning control system," *IEEE Trans. Automatic Control* (Short Papers), vol. AC-15, no. 3, June 1970 (to be published).
[41] K. S. Narendra and D. N. Streeter, "A self-organizing control system based on correlation techniques and selective reinforcement," Cruft Lab., Harvard University, Cambridge, Mass., Tech. Rept. 359, July 1962.

E. Bayesian Estimation for Learning Control

[42] J. T. Tou, *Modern Control Theory*. New York: McGraw-Hill, 1964.
[43] J. C. Hsu and W. E. Meserve, "Decision-making in adaptive control systems," *IRE Trans. Automatic Control*, vol. AC-7, pp. 24–32, January 1962.
[44] N. Ula and M. Kim, "An empirical Bayes approach to adaptive control," *J. Franklin Inst.*, vol. 280, September 1965.
[45] Y. Sawaragi, Y. Sumahara, and T. Nakamizo, *Statistical Decision Theory in Adaptive Control Systems*. New York: Academic Press, 1967.
[46] K. S. Fu, "A class of learning control systems using statistical decision functions," *Proc. 1965 IFAC Symp. of Theory on Self-Adaptive Control Sys.* (Teddington, England).
[47] N. Abramson and D. Braverman, "Learning to recognize patterns in a random environment," *IRE Trans. Information Theory*, vol. IT-8, pp. 58–63, September 1962.
[48] J. Spragins, "A note on the iterative application of Bayes' rule," *IEEE Trans. Information Theory*, vol. IT-11, pp. 544–549, October 1965.
[49] D. G. Keehn, "A note on learning for Gaussian properties," *IEEE Trans. Information Theory*, vol. IT-11, pp. 126–132, January 1965.
[50] H. Teicher, "Identifiability of finite mixtures," *Ann. Math. Statist.*, vol. 34, December 1963.
[51] S. C. Fralick, "Learning to recognize patterns without a teacher," *IEEE Trans. Information Theory*, vol. IT-13, pp. 57–64, January 1967.
[52] J. Spragins, "Learning without a teacher," *IEEE Trans. Information Theory*, vol. IT-12, pp. 223–230, April 1966.
[53] Y. T. Chien and K. S. Fu, "On Bayesian learning and stochastic approximation," *IEEE Trans. Systems Science and Cybernetics*, vol. SSC-3, pp. 28–38, June 1967.
[54] V. S. Pugachev, "Statistical theory of automatic learning systems," *Engrg. Cybernetics*, vol. 5, November–December 1967.

F. Stochastic Approximation

[55] Z. J. Nikolic and K. S. Fu, "A mathematical model of learning in an unknown random environment," *Proc. 1966 NEC.*
[56] ——, "An algorithm for learning without external supervision and its application to learning control systems," *IEEE Trans. Automatic Control*, vol. AC-11, pp. 414–422, July 1966.
[57] Ya. Z. Tsypkin, "Self-learning—what is it?" *IEEE Trans. Automatic Control*, vol. AC-13, pp. 608–612, December 1968.
[58] ——, "Adaptation, learning and self-learning in automatic control systems," *Proc. 1966 IFAC Cong.* (London, England), June 1966, also *Avtomat. i Telemekh.*, vol. 27, no. 1, 1966.
[59] G. N. Saridis, Z. J. Nikolic, and K. S. Fu, "Stochastic approximation algorithms for system identification, estimation, and decomposition of mixtures," *IEEE Trans. Systems Science and Cybernetics*, vol. SSC-5, pp. 8–15, January 1969.
[60] J. D. Hill and K. S. Fu, "A learning control system using stochastic approximation for hill-climbing," *Proc. 1965 JACC* (Troy, N.Y.), pp. 334–340.
[61] E. M. Vaysbard and D. B. Yudin, "Multiextremal stochastic approximation," *Engrg. Cybernetics*, no. 5, pp. 1–11, September–October 1965.
[62] Y. T. Chien and K. S. Fu, "Stochastic learning of time-varying parameters in random environment," *IEEE Trans. Systems Science and Cybernetics*, vol. SSC-5, pp. 237–246, July 1966.

G. Stochastic Automata

[63] G. D. Bruce and K. S. Fu, "A model for finite-state probabilistic systems," *Proc. 1st Allerton Conf. on Circuit and System Theory*, Ann Arbor, Mich.: University of Michigan Press, 1963.
[64] J. W. Carlyle, "Equivalent stochastic sequential machines," Electronics Research Lab., University of California, Berkeley, Tech Rept. ser. 60, issue 415, 1961.
[65] M. O. Rabin, "Probabilistic automata," *Inform. and Control*, vol. 6, pp. 230–245, 1963.
[66] A. Paz, "Some aspects of probabilistic automata," *Inform. and Control*, vol. 9, no. 1, pp. 26–60, 1966.
[67] K. S. Fu and R. W. McLaren, "An application of stochastic automata to the synthesis of learning systems," School of Elec. Engrg., Purdue University, Lafayette, Ind., Tech. Rept. TR-EE-65-17, September 1965.
[68] G. J. McMurtry and K. S. Fu, "A variable structure automaton used as a multimodal searching technique," *IEEE Trans. Automatic Control*, vol. AC-11, pp. 379–387, July 1966.
[69] K. S. Fu and G. J. McMurtry, "A study of stochastic automata as models of adaptive and learning controllers," Purdue University, Lafayette, Ind., Tech Rept. TR-EE-65-17, June 1965.
[70] V. I. Varshavsky and I. P. Vorontsova, "On the behavior of stochastic automata with variable structure," *Avtomat. i Telemekh.*, vol. 24, no. 3, 1963.
[71] M. L. Tsetlin, "On the behavior of finite automata in random environments," *Avtomat. i Telemekh.*, vol. 22, no. 10, 1961.
[72] V. Y. Krylov, "On one automaton that is asymptotically optimal in a random medium," *Avtomat. i Telemekh.*, vol. 24, September 1963.
[73] B. Chandrasekaran and D. W. C. Shen, "On expediency and convergence in variable structure automata," presented at the 5th Symp. on Discrete Adaptive Processes, Chicago, Ill., October 3–5, 1966.
[74] R. W. McLaren, "A stochastic automaton model for the synthesis of learning systems," *IEEE Trans. Systems Science and Cybernetics*, vol. SSC-2, pp. 109–114, December 1966.
[75] K. S. Fu, "Stochastic automata as models of learning systems," in *Computer and Information Sciences*, vol. 2, J. T. Tou, Ed. New York: Academic Press, 1967.
[76] R. W. McLaren, "A stochastic automaton model for a class of learning controllers," *Proc. 1967 JACC* (Philadelphia, Pa.), pp. 267–274.
[77] K. S. Fu and T. J. Li, "Formulation of learning automata and automata games," *Inform. Sci.*, vol. 1, July 1969.
[78] B. Chandrasekaran and D. W. C. Shen, "Stochastic automata games," *IEEE Trans. Systems Science and Cybernetics*, vol. SSC-5, pp. 145–149, April 1969.

[79] J. S. Riordon, "An adaptive automaton controller for discrete-time Markov processes," *Proc. 1969 JACC* (Boulder, Colo.).

H. Miscellaneous

[80] R. Barron, "Self-oganizing control," *Control Engrg.*, February-March 1968.

[81] J. M. Mendel, "Applications of artificial intelligence techniques to a spacecraft control problem," Douglas Aircraft, Santa Monica, Calif., Tech. Rept. DAC-59328, 1966.

[82] M. Garden, "Learning control of valve actuators in direct digital control systems," *Proc. 1967 JACC* (Philadelphia, Pa.), pp. 58-70.

[83] G. K. Krug and A. V. Netushil, "Automatic systems with learning elements," *Proc. 1963 IFAC Cong.* (Basel, Switzerland).

[84] A. Z. Ivanov, G. K. Krug, *et al.*, "Learning-type control systems," *Proc. Moscow Power Institute*, no. 44.

[85] A. V. Netushil, G. K. Krug, and E. K. Letskii, "Use of learning systems for the automation of complex production processes," *Izv. VUZOV SSSR, Mashinostroyenie*, no. 12, 1961.

[86] V. A. Jacubovich, "On adaptive (self-learning) systems of some class," *Proc. 1969 IFAC Cong.* (Warsaw, Poland).

[87] A. G. Ivakhnenko, "Heuristic self-organization in problems of engineering cybernetics," presented at 4th IFAC Cong., Warsaw, Poland, June 16-21, 1969.

[88] E. G. Henrichon, Jr., and K. S. Fu, "Calamity detection using nonparametric statistics," *IEEE Trans. Systems Science and Cybernetics*, vol. SSC-5, pp. 150-155, April 1969.

[89] K. S. Fu and G. A. Ackerson, "Control and estimation in Markov switching environments," School of Elec. Engrg., Purdue University, Lafayette, Ind., Tech. Rept. TR-EE-68-18, August 1968.

[90] M. D. Mesarovic, "On self-organizational systems," in *Self-Organizing Systems—1962*, M. C. Yovits, G. T. Jacobi, and G. D. Goldstein, Eds. Washington, D.C.: Spartan, 1962.

[91] N. D. Naplatanoff, M. P. Marinov, and I. V. Guevski, "Optimal control designed on the principles of learning pattern-recognition systems," *Cybernetica*, vol. 11, no. 1, 1968.

[92] K. Nakamura and M. Oda, "Fundamental principle and behavior of learntrols," in *Computer and Information Sciences*, vol. 2, J. T. Tou, Ed. New York: Academic Press, 1967.

[93] E. M. Braverman and L. I. Rozonoer, "Convergence of random processes in learning machines theory—pt. I," *Avtomat. i Telemekh.*, vol. 30, pp. 57-77, January 1969.

[94] L. A. Zadeh, "Fuzzy sets," *Inform. and Control*, vol. 8, June 1965.

[95] W. G. Wee and K. S. Fu, "A formulation of fuzzy automata and its application as a model of learning systems," *IEEE Trans. Systems Science and Cybernetics*, vol. SSC-5, pp. 215-223, July 1969.

[96] H. Harai, K. Asai, and S. Kitajima, "Fuzzy automaton and its application to learning control systems," *Mem. of the Fac. of Engrg.*, Osaka University, Osaka, Japan, vol. 10, pp. 67-73, 1968.

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

[98] H. D. Gilbert and G. N. Saridis, "Self-organizing solution of the stochastic fuel regulator problem" (submitted to *IEEE Trans. Systems Science and Cybernetics*).

[99] R. A. Jarvis, "Adaptive global search in a time-variant environment using a probabilistic automaton with pattern recognition supervision," *Rec. 1969 IEEE Systems Science and Cybernetics Conf.* (Philadelphia, Pa.).

[100] Y. S. Chow and H. Robbins, "On optimal stopping rules," *Z. Wahrscheinlichkeitstheorie*, vol. 2, 1963.

[101] S. Ye. Rohava, "A discrete device for stochastic approximation," *Soviet Automatic Control*, no. 6, pp. 52-54, November-December 1968.

[102] D. Michie and R. A. Chambers, "Boxes: an experiment in adaptive control," in *Machine Intelligence*, vol. 2, E. Dale and D. Michie, Eds. Edinburgh, Scotland: Oliver and Boyd Ltd., 1968.