Apr 1st, 8:00 AM

Learning Control Systems -Review and Outlook

King-sun Fu
Purdue University

LEARNING CONTROL SYSTEMS - REVIEW AND OUTLOOK*

K. S. Fu
Purdue University
Lafayette, Indiana

Summary

The basic concept of learning control is introduced. The following four learning schemes are briefly reviewed: (1) trainable controllers using linear classifiers, (2) reinforcement learning control systems, (3) Bayesian estimation, and (4) stochastic approximation. Potential applications and problems for further research in learning control are outlined.

1. 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 only be described statistically, for example, in terms of probability distribution or density functions, then stochastic or statistical design techniques will be used. However, if the a priori information required is unknown or incompletely known, in general, an optimal design can not be achieved. 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 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 will be determined on the basis of the estimated information. In the first case, a rather conservative design criterion (for example, Minimax criterion) is ofter 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 to the optimal controller. Here the optimal controller means that the performance of the controller designed will be as equally good as if in the case that 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. The controller learns the unknown information during operation and the learned information is, in turn, used as an experience for future decisions or controls.

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 obtains more information about the unknown function or parameters, the control law will be altered based on the updated information in order to improve the system's performance. A basic block diagram for a learning control system is shown in Figure 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 suboptimal 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 overall system's 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 (i) learning with external supervision (or training or supervised or off-line learning) and (ii) learning without external supervision or online 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). The second approach is that the controller uses a performance measure to direct the learning process (performance feedback approach). 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 between various classes of control situations and the admissible control actions respectively.

*This work was supported by the National Science Foundation Grant, GK-1970.

2. Pattern Classification

Since the problem of classifying different classes of control situations is important in 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 $x^1, x^2, \ldots, x^k$, 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 $\omega_1, \omega_2, \ldots, \omega_m$. 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. The 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 surface can be expressed in terms of "discriminant functions". Associated with each ω_i a discriminant function $d_i(X)$, $i = 1, \ldots, m$ is selected such that if X is from class ω_i then

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

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

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

There are many ways for selecting $d_i(X)$. Several important discriminant functions are discussed in the following [1]

1) Linear discriminant function - The discriminant function $d_i(X)$ is selected as a linear function of feature measurements $x^1, x^2, \ldots, x^k$, i.e.,

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

The decision surface represented by the equation

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

is also a linear function of x^r's or, in other words, a hyperplane in the space Ω_x. Let

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

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 Figure 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, 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 Figure 2, an m-class classifier can be implemented as shown in Figure 3.

2) Polynomial discriminant function - The discriminant function is selected as an n-th order (n > 1) polynomial of $x^1, x^2, \ldots, x^k$. 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 $N = k + \frac{k(k-1)}{2} + k = \frac{k(k+3)}{2}$

let $A = [a_{ij}]$

where $a_{jj} = w_{jj}$, $j=1, \ldots, k$

$$a_{jq} = \frac{1}{2} w_{jq}, \quad j, q = 1, \ldots, k, \; j \neq q$$

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

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

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 hyper-elipsoid.

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

$$d_i(X) = P(\omega_i)\; p(X/\omega_i), \quad i=1, \ldots, m \tag{8}$$

where $P(\omega_i)$ is the a priori probability of class ω_i, and $p(X/\omega_i)$ is a multi-variate conditional density function of X given $X \sim \omega_i$. The decision rule for classifying pattern classes using (8) as the discrimanant function corresponds to the Bayes' decision rule with zero-one loss function in the statistical decision theory[2]. A block diagram for this type of pattern classifier is shown in Figure 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 [3,4]. 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[3]. After each feature measurement is taken, compare the sequential probability ratio

$$\lambda_k = \frac{p_k(X/\omega_1)}{p_k(X/\omega_2)}, \quad k = 1, 2, \ldots \tag{9}$$

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.

$$A = \frac{1-\epsilon_{21}}{\epsilon_{12}} \qquad B = \frac{\epsilon_{21}}{1-\epsilon_{12}}$$

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 classification. If the maximum number of features, N, available is prespecified, the sequential classification procedure must be either truncated at the Nth measurement[4] or a backward computation procedure such as dynamic programming must be used[5]. If all the information required in (3), (6), (8) or (9) 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}'s in (3) and (6) and the $p(X/\omega_i)$'s in (8) and (9) 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.

3. Trainable Controllers

The linear classifier shown in Figure 2 has been used as a trainable controller to realize a switching surface for time-optimal control systems[6,7]. 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 is corresponding to the decision boundary in feature space. The partitioned regions in state space (feature space) correspond to various control situations (pattern classes). Once the desired switching surface (decision boundary) 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 non-linear, the linear classifier used for the controller is a piece-wise linear approximation of the non-linear 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 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 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 multiple number of "1" elements in the pattern representations. Similar codes can be defined with -1, +1 elements instead of 0,1 elements.

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)

TABLE I

The pattern representations (vectors) of the single-spot code are linearly independent 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 that[6] when the state variables are encoded as described, a single linear classifier as shown in Figure 5 will approximate to an arbitrary degree of accuracy (by increasing the number of quantum zones) switching surfaces of the form

$$f(x^1, x^2, \ldots, x^k) = 0$$

provided that no cross-product terms are included in the expression.*

Learning capability is accomplished by the

*Cross-product terms can be realized by using augmented linear classifiers[6].

adjustable weights $w_1, w_2, \ldots, w_N, w_{N+1}$. Refer to Figure 5, the input is the k-dimensional state vector X which is transformed into the N-dimensional vector $[v^1, v^2, \ldots, v^N]^T$. Let

$$V = [v^1, v^2, \ldots, v^N, +1]^T \tag{10}$$

and

$$W = [w_1, w_2, \ldots, w_N, w_{N+1}]^T \tag{11}$$

The output is

$$u = \begin{cases} +1 & \text{if } f(V) > 0 \\ -1 & \text{if } f(V) < 0 \end{cases} \tag{12}$$

where

$$f(V) = V^T W \tag{13}$$

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 space whose optimal control actions u* are known. Specifically, those points in the state space lie on the optimal trajectory X*(t). The points, when transformed into the new space Ω_V, define a training set $T = \{V_j, u_j^*\}$, $j = 1, \ldots, L$. 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

$$\begin{aligned} V^T W &> 0 \quad \text{for each } V \in T_1 \\ \text{and} \quad V^T W &< 0 \quad \text{for each } V \in T_2 \end{aligned} \tag{14}$$

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, (Figure 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:

(A) Least-mean-square-error training procedure - The value of α is

$$\alpha = \frac{|\beta \epsilon|}{V^T V} \tag{15}$$

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

$$\overline{\epsilon^2} = \frac{1}{L} \sum_{j=1}^{L} (d_j - V_j^T W)^2$$

where V_j represents the j-th training pattern vector and d_j the desired binary 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 switching surface.

(B) 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, for any $V \in T_1$, $V^T W > 0$, if the output is erroneous (i.e., $V^T W < 0$) or undefined (i.e., $V^T W = 0$), then let the new weight vector be

$$W' = W + \alpha V \tag{16}$$

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

$$W' = W - \alpha V \tag{17}$$

Before training, W may be preset to any convenient values. Three rules for choosing α are suggested:[1]

(i) Fixed increment rule - α is any fixed positive number.

(ii) Absolute correction rule - α = the smallest integer greater than

$$\frac{|V^T W|}{V^T V} \tag{18}$$

(iii) Fractional correction rule -

$$\alpha = \lambda \frac{|V^T W|}{V^T V}, \quad 0 < \lambda \leq 2 \tag{19}$$

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.

4. 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 responses $\omega_1, \ldots, \omega_m$ are considered. Let $P(\omega_i)$ be the probability of occurence of the i-th class of responses. We consider the performance change being expressed by the change or reinforcement of the set of probabilities $\{P(\omega_i)\}$. Mathematically, the reinforcement of $\{P(\omega_i)\}$ can be described as the following relationship[3, 11].

$$P_{n+1}(\omega_i/X) = \alpha P_n(\omega_i/X) + (1-\alpha)\lambda_n(X;\omega_i) \tag{20}$$

where $P_n(\omega_i/X)$ is the probability of ω_i at instant n given the input X being observed, $0 < \alpha < 1$, $0 \leq \lambda_n(X;\omega_i) \leq 1$ and $\sum_{i=1}^{m} \lambda_n(X;\omega_i) = 1$.

Because of the relationship between $P_{n+1}(\omega_i/X)$ and $P_n(\omega_i/X)$ being linear, (20) is often called a linear reinforcement learning algorithm. It can be easily shown that, if $\lambda_n(X;\omega_i) = \lambda(\omega_i)$, then

$$P_n(\omega_i/X) = \alpha^n P_o(\omega_i) + (1-\alpha^n)\lambda(\omega_i) \tag{21}$$

and

$$\lim_{n \to \infty} P_n(\omega_i/X) = \lambda(\omega_i) \tag{22}$$

It is noted that, from (22), $\lambda(\omega_i)$ is the limiting probability of $P_n(\omega_i/X)$. Hence, $\lambda_n(X;\omega_i)$ should be, in general, related to the information or performance evaluated from the input X at instant n. In learning control system, the input X to the learning controller is usually the output of the plant and ω_i may directly represent the i-th control action. $\lambda_n(X;\omega_i)$ can be identified as the normalized index of performance associated with the i-th class of responses (control actions) of the controller. In some simpler cases, $\lambda_n(X;\omega_i)$ may be 0 or 1 to indicate whether the performance of the system at instant n, due to the i-th control action, is satisfactory or unsatisfactory. Or $\lambda_n(X;\omega_i)$ may be 0 or 1 to indicate whether or not the decision (or classification) ω_i made by the controller at instant n from the input X is correct. In these cases, it can be proved that $P_n(\omega_i/X)$ will converge to its maximum as $n \to \infty$ in the mean and in probability if the i-th control action is a desired one[11,13].

The linear reinforcement learning algorithm has been applied to control systems design[11,15]. In the design of a reinforcement learning controller, the possible classes of response $\omega_i (i = 1,\ldots,m)$ 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. 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 Figure 6.

Waltz and Fu[14] have simulated a class of reinforcement learning control systems on a hybrid computer facility (GEDA-IBM 1620). The feature vector X is essentially the same as the state vector of the plant in this case. The index of performance of the system is of the form

$$IP = \sum_{n=1}^{N} n(x_n^1)^2 \ , \quad x_n^1 = x^1 \text{ at instant } nT \qquad (23)$$

where T 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 $\{u^1, u^2,\ldots,u^m\}$ 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) is chosen as

$$IPS(n) = X_n^T \, G \, X_n \qquad (24)$$

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 having 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 closed 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 algorithm.

Let $P_n(u^i/S_j)$ be the probability that u^i is the best control action for the control situation S_j (or the j-th sample set) at instant nT. Initially, assuming no a priori knowledge, all $P_o(u^i/S_j) = \frac{1}{m}$. $P(u^i/S_j)$ will then be modified according to the following reinforcement algorithm:

$$P_{n+1}(u^i/S_j) = \alpha \, P_n(u^i/S_j) + (1-\alpha) \, \lambda_n(S_j,u^i) \qquad (25)$$

where $\lambda_n(S_j,u^i)$ assumes either 1 or 0 depending upon whether or not the IPS(n) defined in (24) is reduced by applying u^i. α is called learning parameter. The larger α is, the slower the probabilities $P(u^i/S_j)$ converge, which 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, $P(u^i/S_j)$ approaches 1 for u^i and each S_j with the possible exception of those sample sets (control situations) located on the decision surfaces (or called switching boundaries). A control action u^i is used for control situation S_j with probability $P(u^i/S_j)$(a pure random strategy) unless some $P(u^i/S_j)$ exceeds a preset threshold. In this case, the u^i for which $P(u^i/S_j)$ is maximum is used as the control action for S_j.

As learning progresses, most of the probabilities $P(u^i/S_j)$ 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, and $P(u^i/S_j)$ 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 Figure 7. A sampling period T = 0.5 sec. was used. A typical learning curve for the system is shown in Figure 8. Reasonable performance can be obtained for most stationary systems by applying this subset-partition criterion. A second scheme which can be used for boty stationary and nonstationary systems, utilizes the curvature of the approximated (learned) switching boundary to determine where subsets should be established. The utilization of a priori knowledge for more efficient partition and the problem of subgoal selection has recently been studied by Jones[16,17]. The chain encoding scheme described by Freeman[18] 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.

5. Bayesian Learning in Control Systems

In the statistical design of an optimal controller using dynamic programming[19] or statistical decision theory[20-22], 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

$$X_{n+1} = g(X_n, u_n) \tag{26}$$

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 $I_n = E[\sum_{n=1}^{N} F(X_n, u_{n-1})]$, a recurrence relationship can be derived using dynamic programming with the probability density function p(X) known[19]. Similar to the case mentioned in statistical pattern classification, if these 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[23]. 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' theorem to estimate the unknown information is introduced in this section[23-26].

Suppose that the probability density function $P(X/\omega_i)$ is to be learned, where ω_i represents the i-th class of control situations. Let $X_1, \ldots, X_n$ 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 $X_1, \ldots, X_n$. 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.,

$$p(\theta/\omega_i, X_1, \ldots, X_n) = \frac{p(X_n/\omega_i, \theta, X_1, \ldots, X_{n-1})\, p(\theta/\omega_i, X_1, \ldots, X_{n-1})}{p(X_n/\omega_i, X_1, \ldots, X_{n-1})} \tag{27}$$

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, $p_o(\theta/\omega_i)$, be also Guassian distributed with initial mean vector M_o and initial covariance matrix Φ_o. Then, after the first sample measurement X_1 has been taken

$$p(\theta/\omega_i, X_1) = \frac{p(X_1/\omega_i, \theta)\, p_o(\theta/\omega_i)}{p(X_1/\omega_i)} \tag{28}$$

It is noted that the assumption of a Gaussian distribution for $p_o(\theta/\omega_i)$ will simplify the computation of (28) since the product of $p(X_1/\omega_i, \theta)$ $p_o(\theta/\omega_i)$ is also a Gaussian distribution. By using this property of reproducible distribution of $p_o(\theta/\omega_i)$ and the iterative application of Bayes' theorem, after n learning samples, a recursive expression for estimation θ=M is given as[24]

$$M_n = K(\Phi_{n-1}+K)^{-1} M_{n-1} + \Phi_{n-1}(\Phi_{n-1}+K)^{-1} X_n \tag{29}$$

and

$$\Phi_n = K(\Phi_{n-1} + K)^{-1}\, \Phi_{n-1} \tag{30}$$

In terms of the initial estimates M_o and Φ_o, (29) and (30) becomes

$$M_n = (n^{-1}K)(\Phi_o + n^{-1}K)^{-1} M_o + \Phi_o(\Phi_o + n^{-1}K)^{-1} < K > \tag{31}$$

and

$$\Phi_n = (n^{-1} K)(\Phi_o + n^{-1} K)^{-1}\, \Phi_o \tag{32}$$

where $< X > = \frac{1}{n} \sum_{i=1}^{n} X_i$ is the sample mean. Equation (31) shows that the n-th estimate of the mean vector, M_n, can be interpreted as a weighted average of the a priori mean vector M_o and the sample information $< X >$. As $n \to \infty$, $M_n \to < X >$ 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[25].

If the correct classifications of the learning samples $X_1, \ldots, X_n$ are not available, a nonsupervised learning technique must be used. In this case, each measurement X_i may be considered as 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.,

$$p(X/\theta, P) = \sum_{i=1}^{m} P(\omega_i)\, p(X/\omega_i, \theta) \tag{33}$$

where θ_i is the unknown parameter associated with $p(X/\Phi_i)$, and $\theta = \{\theta_i;\ i = 1, \ldots, m\}$, $P = \{P(\omega_i);\ i = 1, \ldots, m\}$. Let $B = (\theta, P)$ and consider that the sequence of independent measurements $X_1, \ldots, X_n$ are taken from the mixture with probability density function p(X). Then a successive application of Bayes' theorem given

$$p(B/X_1, \ldots, X_n) = \frac{p(X_n/X_1, \ldots, X_{n-1}, B)\, p(B/X_1, \ldots, X_{n-1})}{p(X_n/X_1, \ldots, X_{n-1})} \tag{34}$$

It is necessary to select the a priori probability $p_o(B)$ 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: The mixture $p(X/\theta, P)$ is said to be identifiable[27] if the mapping of θ and P onto $p(X/\theta, P)$, defined by (33), is a one-to-one mapping. Note that the question of whether $p(X/\theta, P)$ is identifiable or not is one of unique characterization. That is, for a particular family of the i-th component (parameter 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 $\{P(\omega_i)\}$. It is then clear that if the nonsupervised learning problem is such that the mixture is not uniquely characterized by $\{\theta_i\}$ and $\{P(\omega_i)\}$ (not identifiable), then there exists no unique solution to the underlying estimation problem. In addition to Bayesian learning technique[26] the stochastic approximation procedure discussed in Section 6 can also be applied for estimating unknown parameters in a mixture distribution[28-30].

6. Learning Control Systems Using Stochastic Approximation

The learning control systems discussed in Section 4 and Section 5 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[30-32]. 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's 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 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

$$y_{n+1} = \Phi_{n+1}(y_n, u_{n+1}) \tag{35}$$

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

$$z_{n+1} = f(y_{n+1}, u_{n+1}, y_n) \tag{36}$$

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.,

$$p(z_{n+1}; u_n = u^r, y_n = y, u_{n+1} = u^j)$$
$$= p(z/u^r, y, u^j) \tag{37}$$

for every n. The performance index of the system is

$$IP = E[z/u^r, y, u^j] \tag{38}$$

The optimal control action u* is defined by

$$E[z/u^r, y, u^*] = \underset{j=1,\dots,m}{\mathrm{Min}} \{E[z/u^r, y, u^j]\} \tag{39}$$

Since $p(z/u^r,y,u^j)$, $j = 1,\dots, m$ and Φ_n are unknown, $E[z/u^r,y,u^j]$ can only be obtained from the successive estimates $\hat{E}_{N_j}[z/u^r,y,u^j]$, $N_j = 1,2,\dots$ 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

$$E[z/u^r, y, u^j] = E[z/X^q, u^j] \tag{40}$$

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 occurence of X^q. The stochastic approximation procedure is used to estimate $E[z/X^q, u^j]$, i.e.,

$$\hat{E}_{N_{qj}+1}[z/X^q,u^j] = \hat{E}_{N_{qj}}[z/X^q,u^j]$$
$$+ \gamma_{N_{qj}}\{z_{N_{qj}+1} - \hat{E}_{N_{qj}}\{z/X^q,u^j]\} \tag{41}$$

for $N_{qj} = 0,1,2,\dots$, where $\hat{E}_0[z/X^q,u^j] = 0$ and $\gamma_{N_{qj}} = 1/N_{qj}$. Then

$$P\{\lim_{N_{qj}\to\infty} \hat{E}_{N_{qj}}[z/X^q,u^j]$$
$$= E[z/X^q, u^j]\} = 1 \tag{42}$$

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

$$P(u^*/X^q) = 1 \tag{43}$$

The subjective probabilities $\{P_{n_q}(u^k/X^q);\ k = 1,\dots,m\}$ 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. 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 $\hat{E}_{n_q+1}[z/X^q,u^j]$, k = 1,...,m. The subjective probabilities are recursively computed for every u^k, $k = 1,\dots,m$, by

$$P_{n_q+1}(u^k/X^q) = P_{n_q}(u^k/X^q) + \gamma_{n_q+1}[\xi_{n_q+1}(X^q;u^k)$$
$$- P_{n_q}(u^k/X^q)] \tag{44}$$

where (i)

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

and

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

and (ii) $\xi_{n_q+1}(X^q;u^k)$

$$= \begin{cases} 1 \text{ if } \hat{E}_{n_q+1}[z/X^q,u^k] = \mathrm{Min}_j\ \hat{E}_{n_q+1}[z/X^q,u^j] \\ 0 \text{ if } \hat{E}_{n_q+1}[z/X^q,u^k] \neq \mathrm{Min}_j\ \hat{E}_{n_q+1}[z/X^q,u^j] \end{cases} \tag{45}$$

It can be shown that if, for every suboptimal control action u^u [30],

$$\sum_{n_q=1}^{\infty} \gamma_{n_q} E[\xi_{n_q}(X^q;u^u)/z_1,\ldots,z_{n_q}] < \infty \qquad (46)$$

then

$$P\{\lim_{n_q \to \infty} P_{n_q}(u^*/X^q) = 1\} = 1 \qquad (47)$$

Equation (47) indicates that the desired optimal control law as defined in (43) will be eventually obtained with probability one.

7. Conclusions and Remarks

The basic concept of learning control has been reviewed. Several important learning techniques have been described. Theoretically speaking, these techniques have similar learning properties[33-35]. However, from an engineering viewpoint, the a priori information required and the computation involved for these techniques are different. Recently, stochastic automata with variable structures have been proposed as models for learning systems. Simple applications have been made on pattern recognition and learning control systems[36,37].

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, and any additional a priori information (for example, the form of the plant equation, the type of the environmental disturbance, etc.) will improve the learning rate of the system. 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.

Learning control is a new area of research. Preliminary attempts of applying theoretical results to spacecraft control problems have already been made by several authors[15,38-40]. Other applications include the control of valve actuators[41], the control of power systems and production processes [42-44]. 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 the theoretical study, many problems, for example, new algorithms with higher learning, the determination of proper stopping rules and learning in nonstationary environments, still need to be solved.

References

1. Nilsson, N. J., "Learning Machines". McGraw-Hill, New York, 1965.

2. Chow, C. K., An optimum character recognition system using decision functions. IRE TRANS. on Electronic Computers, Vol. EC-6, pp.247-254, December (1957).

3. Fu, K. S., A sequential decision model for optimum recognition. Biological Prototype and Synthetic Systems, Vol. I, Plenum Press, (1962).

4. Chien, Y. T., and Fu, K. S., A modified sequential recognition machine using time-varying stopping boundaries. IEEE Trans. on Information Theory, Vol. IT-12, April (1966).

5. Fu, K. S., Chien, Y. T., and Cardillo, G. P., A dynamic programming approach to sequential pattern recognition. IEEE Trans. on Electronic Computers, (1967).

6. Smith, F. W., Contact control by adaptive pattern-recognition techniques. Tech. Rept. No. 6762-2. Stanford Electronic Laboratories, Stanford University, California, April 1964.

7. Widrow, B., and Smith, F. W., Pattern recognizing control systems. Computer and Information Sciences, ed., J. T. Tou and R. H. Wilcox, Spartan Books, (1964).

8. Bush, R., and Mosteller, F., "Stochastic Models for Learning". John Wiley and Sons, 1955.

9. Tsetlin, M. L., On the behavior of finite Automata in random environments. Avtomatika i Telemekhanika, Vol. 22, No. 10, (1961).

10. Varshavsky, V. I., and Vorontsova, I. P., On the behavior of stochastic automata with variable structure. Avtomatika i Telemekhanika, Vol. 24, No. 3, (1963).

11. Fu, K. S., and McLaren, R. W., An application of stochastic automata to the synthesis of learning systems. Tech. Rept. TR-EE-65-17. School of Electrical Engineering, Purdue University, September 1965.

12. McMurty, G. J., and Fu, K. S., A variable structure automaton used as a multi-model searching technique. IEEE Trans on Automatic Control, Vol. AC-11, July (1966).

13. Fu, K. S., and Nikolic, Z. J., On some reinforcement techniques and their relations with stochastic approximation. IEEE Trans. on Automatic Control, Vol. AC-11, October (1966).

14. Waltz, M. D., and Fu, K. S., A heuristic approach to reinforcement learning control systems. IEEE Trans. on Automatic Control, Vol. AC-10, October (1965).

15. Mendel, J. M., Survey of learning control systems for space vehicle applications. Preprints, JACC, August (1966).

16. Jones, L. E., III, On the choice of subgoals for learning control systems. IEEE Trans. on Automatic Control, December (1968).

17. Jones, L. E., III, and Fu, K. S., A learning control system--design considerations. Tech. Rept. TR-EE-68-32, School of Electrical Engineering, Purdue University, October 1968.

18. Freeman, H., On the digital computer classification of geometric line patterns. Proc. National Electronics Conference, Vol. 18, October (1962).

19. Tou, J. T., "Modern Control Theory". McGraw-Hill, 1964.

20. Hsu, J. C., and Meserve, W. E., Decision-making in adaptive control systems. Trans. IRE on Automatic Control, pp. 24-32, January (1962).

21. Ula, N., and Kim, M., An empirical Bayes approach to adaptive control. J. Franklin Institute, Vol. 280, No. 3, September (1965).

22. Sawaragi, Y., Sumahara, Y., and Nakamizo, T., "Statistical Decision Theory in Adaptive Control Systems". Academic Press, 1967.

23. Fu, K. S., A class of learning control systems using statistical decision functions.

Proc. Second IFAC (Teddington) Symposium on Theory on Self-Adaptive Control Systems, September (1965).
24. Braverman, D., and Abramson, N., Learning to recognize patterns in a random environment. IRE Trans. on Information Theory, Vol. IT-8, September (1962).
25. Keehn, D. G., A note on learning for Guassian properties. IEEE Trans. on Information Theory, Vol. IT-11, January (1965).
26. Fralick, S. C., Learning to recognize a pattern without a teacher. IEEE Trans. on Information Theory, Vol. IT-13, January (1967).
27. Teicher, H., Identifiability of finite mixtures. Ann. Math. Stat., Vol. 34, December (1963).
28. Chien, Y. T., and Fu, K. S., On Bayesian learning and stochastic approximation. IEEE Trans. on System Science and Cybernetics, June (1967).
29. Nikolic, Z. J., and Fu, K. S., On the estimation and decomposition of mixture using stochastic approximation. Proc. 1967 SWIEECO, April (1967).
30. Nikolic, Z. J., and Fu, K. S., A mathematical model of learning in an unknown random environment. Proc. 1966 National Electronics Conference, October (1966).
31. Nikolic, Z. J., and Fu, K. S., An algorithm for learning without external supervision and its application to learning control systems. IEEE Trans. on Automatic Control, Vol. AC-11, July (1966).
32. Tsypkin, Ya. Z., Adaptation, learning and self-learning in control systems. Third IFAC, London, June (1966).
33. Fu, K. S., Nikolic, Z. J., Chien, Y. T., and Wee, W. G., On the stochastic approximation and related learning techniques. Tech. Rept. TR-EE-66-6. School of Electrical Engineering, Purdue University, April 1966.
34. Fu, K. S., Learning control systems. Advances in Information Systems Science, ed. J. T. Tou, Plenum Press, (1969).
35. Fu, K. S., Learning system theory. Chapter 11, System Theory, ed., L. A. Zadeh and E. Polak, Mc-Graw-Hill, New York, (1969).
36. Fu, K. S., Stochastic automata as models of learning systems. Computer and Information Sciences II, ed., J. T. Tou, Academic Press, (1967).
37. McLaren, R. W., A stochastic automaton model for a class of learning controllers. Preprints, 1967 Joint Automatic Control Conference, (1967).
38. Smith, F. B., Jr., Trainable flight control system investigation. FDL-TDR-64-89, Wright-Patterson AFB, Ohio, 1964.
39. Barron, R., Self-organizing control. Control Engineering, February, March (1968).
40. Mendel, J. M., Applications of artificial intelligence techniques to a spacecraft control problem. Douglas Report DAC-59328, Santa Monica, California (1966).
41. Garden, M., Learning control of valve actuators in direct digital control systems. Preprints, Joint Automatic Control Conference, (1967).
42. Krug, G. K., and Netushil, A. V., Automatic systems with learning elements. Proc. IFAC Congress, (1963).
43. Ivanov, A. Z., Krug, G. K., et. al., Learning-type control systems. Proc. Moscow Power Institute, No. 44.
44. Netushil, A. V., Krug, G. K., and Letskii, E. K., Use of learning systems for the automation of complex production processes. Izv. VUZOV SSSR, Mashinostroyenie, No. 12 (1961).
45. Fu, K. S., Learning control systems. Computer and Information Sciences, ed., J. T. Tou and R. H. Wilcox, Spartan Books, (1964).
46. Sklansky, J., Learning systems for automatic Control. IEEE Trans. on Automatic Control, Vol. AC-11, January (1966).
47. Tou, J. T., and Hill, J. D., Steps toward learning control. Preprints, JACC, August (1966).
48. Fu, K. S., "Sequential Decision Methods in Pattern Recognition and Machine Learning." Academic Press, New York, 1968.
49. Butz, A. R., Learning bang-bang regulators. Proc. Hawaii International Conference on System Sciences, January (1968).
50. Leondes, C. T., and Mendel, J. M., Artificial intelligence control. Douglas Paper No. 4336, McDonnell-Douglas, Santa Monica, California, January 1967.

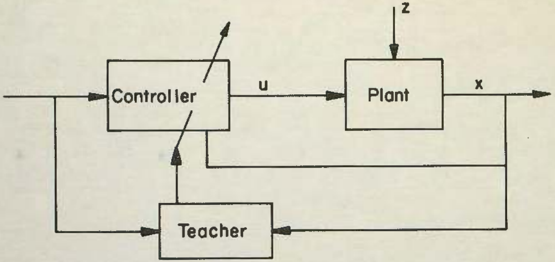

FIGURE 1.

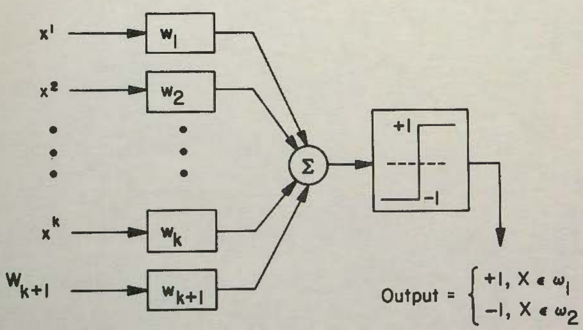

FIGURE 2.

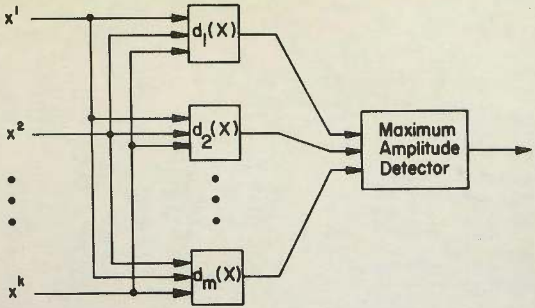

FIGURE 3.

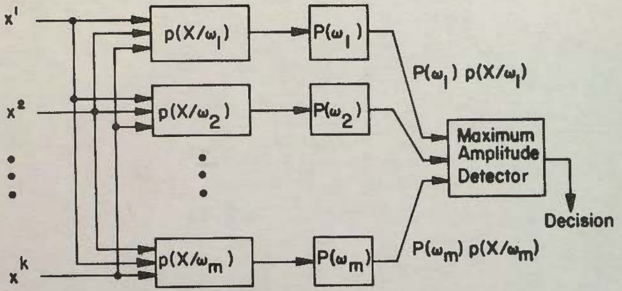

FIGURE 4.

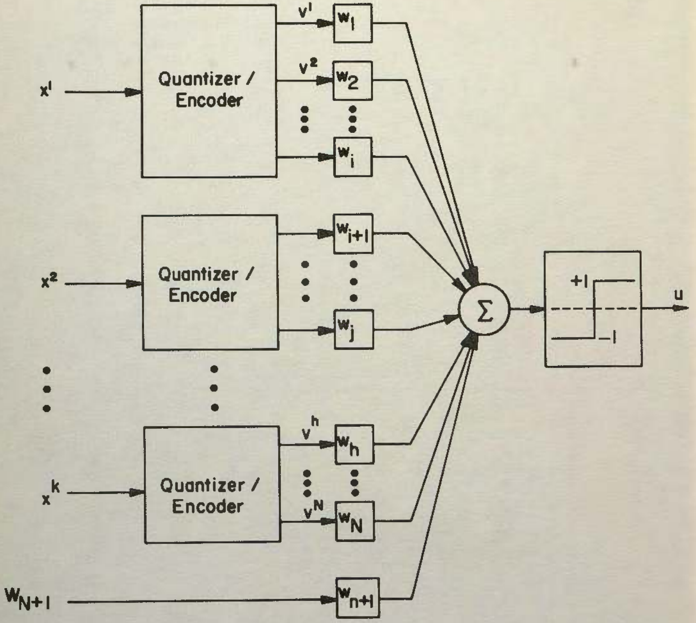
Quantizer / Encoder
Quantizer / Encoder
Quantizer / Encoder
x^1
x^2
x^k
w_{N+1}
v^1
v^2
v^h
v^N
w_1
w_2
w_i
w_{i+1}
w_j
w_h
w_N
w_{n+1}
Σ
+1
−1
u

FIGURE 5.

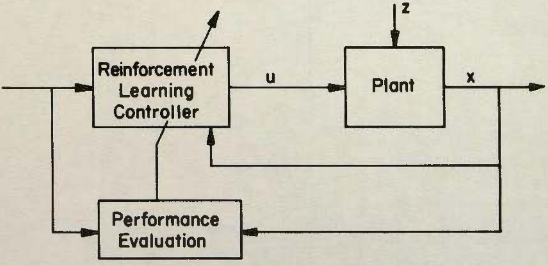
z
Reinforcement
Learning
Controller
u
Plant
x
Performance
Evaluation

FIGURE 6.

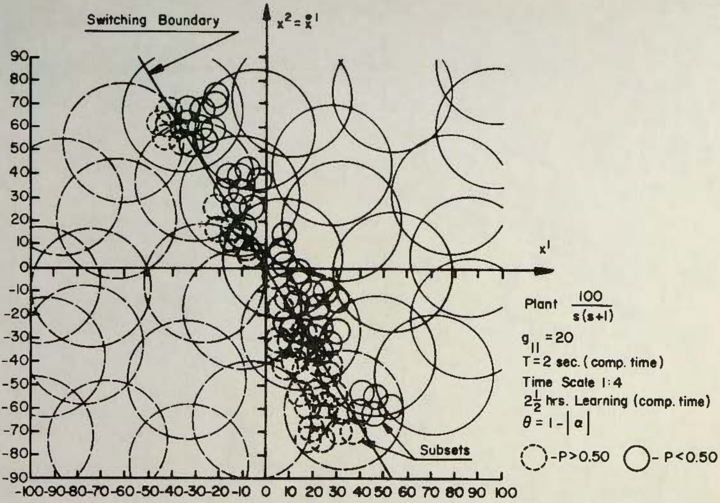
Switching Boundary
$x^2 = \dot{x}^1$
x^1
90
80
70
60
50
40
30
20
10
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100 -90 -80 -70 -60 -50 -40 -30 -20 -10 0 10 20 30 40 50 60 70 80 90 100
Plant $\frac{100}{s(s+1)}$
$g_{11} = 20$
T = 2 sec. (comp. time)
Time Scale 1:4
$2\frac{1}{2}$ hrs. Learning (comp. time)
$\theta = 1 - |\alpha|$
Subsets
-P>0.50
- P<0.50

FIGURE 7.

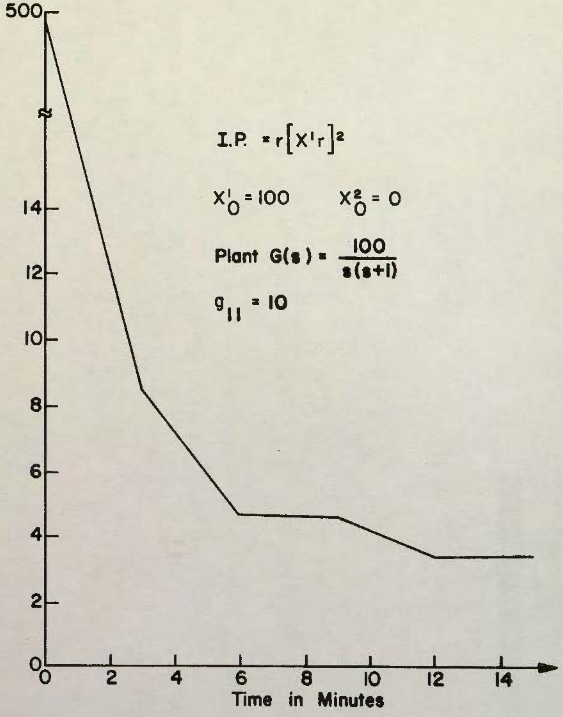
500
14
12
10
8
6
4
2
0
0
2
4
6
8
10
12
14
Time in Minutes
I.P. = r[X'r]²
X¹₀ = 100   X²₀ = 0
Plant G(s) = 100 / s(s+1)
g₁₁ = 10

FIGURE 8.