*Sādhanā*, Vol. 15, Parts 4 & 5, December 1990, pp. 263–281. © Printed in India.

# Stochastic automata and learning systems

M A L THATHACHAR

Department of Electrical Engineering, Indian Institute of Science, Bangalore 560012, India

**Abstract.** We consider stochastic automata models of learning systems in this article. Such learning automata select the best action out of a finite number of actions by repeated interaction with the unknown random environment in which they operate. The selection of an action at each instant is done on the basis of a probability distribution which is updated according to a learning algorithm. Convergence theorems for the learning algorithms are available. Moreover the automata can be arranged in the form of teams and hierarchies to handle complex learning problems such as pattern recognition. These interconnections of learning automata could be regarded as artificial neural networks.

**Keywords.** Stochastic automata; learning systems; artificial neural networks.

## 1. Introduction

Stochastic automata operating in unknown random environments form general models for learning systems. Such models are based on some early studies made by American psychologists (Bush & Mosteller 1958) and also on automata studies introduced into engineering literature by Soviet researchers (Tsetlin 1973). Stochastic automata interact with the environment, gain more information about it and improve their own performance in some specified sense. In this context, they are referred to as learning automata (Narendra & Thathachar 1989).

The need for learning exists in identification or control problems when high levels of uncertainty are present. For instance, the pole balancing problem can be regarded as a deterministic control problem when the relevant parameters of the system such as the mass of the cart and mass and length of the pole are given. When some of the parameters are unknown, it can be treated as an adaptive control problem. When even the dynamic description of the system is not used to determine the control, learning becomes necessary. In the latter approach, the input to the system is generated by experience through a learning algorithm which builds up associations between the input and output.

Learning is very much relevant in tasks such as pattern recognition where detailed mathematical descriptions are usually not available and feature measurements are noisy. Similar is the situation with regard to computer vision problems such as stereopsis.

In recent years there has been a great deal of interest in artificial neural networks. The main aim here is to use a dense interconnection of simple elements to build systems which provide good performance in perceptual tasks. The learning automata described here appear to be well-suited as building blocks of stochastic neural networks as they can operate in highly noisy environments.

## 2. The random environment

The learning model that we consider involves the determination of the optimal action out of a finite set of allowable actions. These actions are performed on a random environment. The environment responds to the input action by an output which is probabilistically related to the input action.

Mathematically the random environment can be defined by a triple $\{\alpha, d, \beta\}$ where

$\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents the input set,

$d = \{d_1, d_2, \ldots, d_r\}$, the set of reward probabilities, and

$\beta = \{0, 1\}$, a binary set of outputs.

The input $\alpha(n)$ to the environment is applied at discrete time $n = 0, 1, 2, \ldots$ etc. and belongs to the set $\alpha$. The output $\beta(n)$ is instantaneously related to the input $\alpha(n)$ and is binary in the simplest case. An output $\beta(n) = 1$ is called a reward or favourable response and $\beta(n) = 0$ is called a penalty or unfavourable response. The reward probabilities relate $\alpha(n)$ and $\beta(n)$ as follows,

$$d_i = P\{\beta(n) = 1 \mid \alpha(n) = \alpha_i\}, \quad (i = 1, \ldots, r). \tag{1}$$

Consequently $d_i$ represents the probability that the application of an action $\alpha_i$ results in a reward output. An equivalent set of penalty probabilities $c_i = 1 - d_i$ can also be defined.

$$c_i = P\{\beta(n) = 0 \mid \alpha(n) = \alpha_i\}, \quad (i = 1, \ldots, r). \tag{2}$$

Models in which the output of the environment is binary are called $P$-models. A generalization of this model is the $Q$-model where the output belongs to a finite set in the interval $[0, 1]$. A further generalization is the $S$-model where the output is a continuous random variable which assumes values in $[0, 1]$. $Q$ and $S$ models provide improved discrimination of the nature of response of the environment. The penalty probabilities are usually regarded as constants. In such a case the environment is called stationary. When $c_i$ vary with time $n$ directly or indirectly, the environment is nonstationary. Many practical problems involve nonstationary environments.

The problem of learning in an unknown random environment involves the determination of the optimal action from the input-output behaviour of the environment. The optimal action is the action most effective in producing the reward output or the favourable response.

Let

$$d_m = \max_i \{d_i\} \tag{3}$$

Then $\alpha_m$ is the optimal action and could be easily identified if each $d_i$ is known. Hence a meaningful learning problem exists only when the reward probabilities are unknown. In some problems the set of reward probabilities $\{d_i\}$ may be known but the action with which each reward probability is associated may not be identified. Learning the optimal action is a relevant problem even here.

Learning involves experimentation on the environment by choosing input actions and correlating them with outputs in developing a strategy for picking new actions. A stochastic automaton is very helpful in carrying out such operations in an effective manner.

## 3. The learning model

The stochastic automata that we consider in the learning model can be represented by the triple $\{\alpha, \beta, A\}$. $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the finite set of outputs or actions; $\beta = \{0, 1\}$ is the binary set of inputs; $A$ is the algorithm for updating the action probability vector $p(n)$.

$$p(n) = [p_1(n), p_2(n), \ldots, p_r(n)]^T, \tag{4}$$

where

$$p_i(n) = P\{\alpha(n) = \alpha_i\},$$

and $T$ denotes transpose. Thus the stochastic automaton selects an output action $\alpha(n)$ at time $n$ randomly based on $p(n)$. It updates $p(n)$ to $p(n+1)$ according to the algorithm $A$ and selects a new action $\alpha(n+1)$.

The learning model that we study consists of the stochastic automaton in a feedback connection with the random environment (figure 1). The action $\alpha(n)$ of the automaton forms the input to the environment and the output $\beta(n)$ of the environment is the input to the automaton. An automaton acting in an unknown random environment so as to improve its performance in some specified sense is called a learning automaton.

This learning model is closely related to the two-armed bandit problem extensively studied in statistics. The two-armed bandit is a machine with two arms. A human test subject is asked to pull one of the 2 arms. The subject is rewarded or punished according to a previously determined schedule with a fixed probability of reward for the two choices. The problem is in finding whether the subject will learn to select the better arm associated with the higher probability of reward in successive experiments.

The above problem illustrates the classical dilemma encountered between identification and control in such learning situations. The subject must decide which of the 2 arms must be chosen on the basis of the previous performance. The dilemma is whether the subject should choose the arm which is known to be better so far or whether he should select the arm about which least knowledge exists so that new knowledge of relative effectiveness of the arms is obtained.

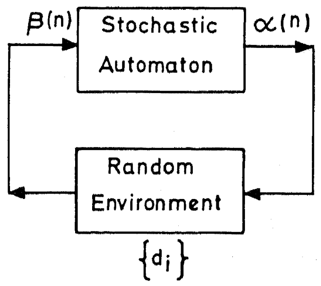

**Figure 1.** Stochastic automaton in random environment.

## 4. Norms for learning

Although learning is understood in a qualitative way, it is necessary to set up quantitative norms for a proper understanding of the model. On close scrutiny it is found that several kinds of learning can be defined. Some prominent definitions are given below.

At the outset it may be noted that the action probability vector $p(n)$ is a random variable and the sequence $\{p(n)\}$ is a random process. A quantity of much importance is the Average Reward $W(n)$ defined as

$$\begin{aligned} W(n) &= E[\beta(n)|p(n)] \\ &= \sum_{i=1}^{r} P[\beta(n)=1|\alpha(n)=\alpha_i]\, P[\alpha(n)=\alpha_i] \\ &= \sum_i d_i p_i(n). \end{aligned} \tag{5}$$

If actions are selected purely randomly (i.e. $p_i = 1/r$ for each $i$) then $W(n) = W_0 = 1/r\,\Sigma_i d_i$. If an automaton is said to learn, it must perform better than this, in which case it is called *expedient.*

DEFINITION 4.1

A learning automaton is said to be *Expedient* if

$$\lim_{n\to\infty} E[W(n)] > W_0. \tag{6}$$

The best behaviour we can expect from the automaton is defined as follows.

DEFINITION 4.2

A learning automaton is said to be *Optimal* if

$$\lim_{n\to\infty} E[W(n)] = d_m, \tag{7}$$

where

$$d_m = \max_i \{d_i\}.$$

An equivalent definition is,

$$\lim_{n\to\infty} p_m(n) = 1 \text{ (with probability 1) (w.p.1).} \tag{8}$$

DEFINITION 4.3

A learning automaton is said to be $\varepsilon$-*optimal* if

$$\lim_{n\to\infty} E[W(n)] > d_m - \varepsilon \tag{9}$$

can be obtained for any arbitrary $\varepsilon > 0$ by a proper choice of the parameters of the automaton.

Another definition closely related to $\varepsilon$-optimality is the following.

DEFINITION 4.4

A learning automaton is said to be *Absolutely Expedient* if

$$E[W(n+1)|p(n)] > W(n) \tag{10}$$

for all $n$, all $p_i(n) \in (0,1)$ and for all possible sets $\{d_i\}$ $(i=1,\ldots,r)$ except those in which all reward probabilities are equal.

It can be seen that absolute expediency implies that $W(n)$ is a submartingale and consequently $E[W(n)]$ monotonically increases in arbitrary environments. Furthermore with some additional conditions it ensures $\varepsilon$-optimality.

## 5. Learning algorithms

The basic operation in a learning automaton is the updating of action probabilities. The algorithm used for this operation is known as the learning algorithm or the reinforcement scheme (Lakshmivarahan 1981). In general terms, the learning algorithm can be represented by,

$$p(n+1) = T[p(n), \alpha(n), \beta(n)], \tag{11}$$

where $T$ is an operator. The algorithm generates a Markov process $\{p(n)\}$. If $p(n+1)$ is a linear function of $p(n)$, the reinforcement scheme is said to be linear. Sometimes the scheme may be characterized by the asymptotic behaviour of the learning automaton using it i.e. expedient, optimal etc.

Let $S_r$ be the unit simplex defined by

$$S_r = \{p \,|\, p = [p_1, p_2, \ldots, p_r]^T, \quad 0 \leqslant p_i \leqslant 1, \sum_{i=1}^{r} p_i = 1\}. \tag{12}$$

Then $S_r$ is the state space of the Markov process $\{p(n)\}$. The interior of $S_r$ is represented as $S_r^0$. Let $e_i = [0, 0, \ldots, 1, 0, 0]^T$ where the $i$th element is unity, be an $r$-vector. Then the set of all $e_i (i = 1, \ldots, r)$ is the set of vertices $V_r$ of the simplex $S_r$.

If the learning algorithm is chosen so that $\{p(n)\}$ has absorbing states, then it is called an absorbing algorithm. Otherwise it is a nonabsorbing algorithm. The two types of algorithms have different types of behaviour.

We shall first consider a few linear algorithms which are simple to implement and analyse.

### 5.1 *Linear reward–penalty* ($L_{R-P}$) *scheme*

This scheme has been extensively treated in the psychology literature (Bush & Mosteller 1958). It can be described as follows.

If $\alpha(n) = \alpha_i$,

$$\begin{aligned} p_i(n+1) &= p_i(n) + a[1 - p_i(n)], \quad \text{if } \beta(n) = 1, \\ p_j(n+1) &= (1-a)\, p_j(n), \\ p_i(n+1) &= (1-b)\, p_i(n), \quad \text{if } \beta(n) = 0, \\ p_j(n+1) &= b/(r-1) + (1-b)\, p_j(n), \end{aligned} \tag{13}$$

where $0 < a < 1$ and $b = a$. Computing the conditional expectation $E[p_i(n+1)|p(n)]$ and rearranging,

$$E[p(n+1) = A^T E[p(n)], \tag{14}$$

where $A$ is an $(r \times r)$ stochastic matrix with elements,

$$a_{ii} = (1 - ac_i) \text{ and } a_{ij} = ac_i/(r-1).$$

$A$ has one eigenvalue at unity and the rest in the interior of the unit circle. The asymptotic solution of (14) can be computed as the eigenvector of $A$ corresponding to unity eigenvalue and is given by

$$\operatorname*{Lim}_{n\to\infty} E[p_i(n)] = (1/c_i) \Big/ \left[\sum_{j=1}^{r} (1/c_j)\right], \quad (i = 1, \ldots, r).$$

It follows that

$$\operatorname*{Lim}_{n\to\infty} E[W(n)] = 1 - \left\{ r \Big/ \left[\sum_{j=1}^{r} (1/c_j)\right]\right\} > \left\{\left(\sum_{j=1}^{r} d_j\right) \Big/ r\right\} = W_0,$$

and hence $\mathrm{L}_{\mathrm{R-P}}$ is expedient in all stationary random environments.

## 5.2 *The linear reward*-inaction ($L_{R-I}$) *scheme*

This algorithm (Shapiro & Nagendra 1969) can be obtained by setting $b = 0$ in (13). It is sometimes called a 'benevolent' scheme as there is no updating under a penalty input.

Let

$$\begin{aligned}\Delta p_i(n) &= E[p_i(n+1) - p_i(n)|p(n)] \\ &= a\, p_i(n) \sum_j p_j(n)(d_i - d_j).\end{aligned} \tag{15}$$

Hence,

$$\Delta p_m(n) = a\, p_m(n) \sum_j p_j(n)(d_m - d_j) > 0 \tag{16}$$

for all $p \in S_r^0$, since $(d_m - d_j) > 0$ for all $j \neq m$.

It follows that $E[p_m(n)]$ is monotonically increasing in any stationary random environment and this is certainly a desirable feature. It can also be checked that each vertex $e_i$ of $S_r$ is an absorbing state of the Markov process $\{p(n)\}$. Furthermore one can show that $\{p(n)\}$ converges to an element of the set of vertices $V_r$ w.p.1 and also that $P\{\mathrm{Lim}_{n\to\infty} p_m(n) = 1\}$ can be made arbitrarily close to unity by selecting sufficiently small '$a$'. This implies that the $\mathrm{L}_{\mathrm{R-I}}$ scheme is $\varepsilon$-optimal in all stationary random environments.

## 5.3 *The linear reward – ε-penalty* ($L_{R-\varepsilon P}$) *scheme*

The $\mathrm{L}_{\mathrm{R-I}}$ scheme has the drawback of being associated with several absorbing states. If the process $\{p(n)\}$ starts at an absorbing state it continues in that state and this may not be desirable. However the scheme is $\varepsilon$-optimal. One can continue to enjoy

$\varepsilon$-optimality without the disadvantage of absorbing states by setting the parameter $b$ in (13) to a small positive number ($b \ll a < 1$). This choice gives the $L_{R-\varepsilon P}$ scheme (Lakshmivarahan 1981).

## 5.4 *Absolutely expedient schemes*

Early studies of reinforcement schemes were made in a heuristic fashion. A synthesis approach led to the concept of absolutely expedient schemes (Lakshmivarahan & Thathachar 1973). The concept arose as a result of the following question: What are the conditions on the functions appearing in the reinforcement scheme that ensure desired behaviour?

The importance of absolutely expedient schemes arises partly from the fact that they represent the only class of schemes for which necessary and sufficient conditions of design are available. They can be considered as a generalization of the $L_{R-I}$ scheme. They are also $\varepsilon$-optimal in all stationary random environments.

Consider a general reinforcement scheme of the following form.

If $\alpha(n) = \alpha_i$,

$$\begin{aligned}
p_j(n+1) &= p_j(n) - g_j(p(n)), \quad \text{when } \beta(n) = 1,\\
p_i(n+1) &= p_i(n) + \sum_{j \neq i} g_j(p(n)), \quad (j \neq i),\\
p_j(n+1) &= p_j(n) + h_j(p(n)), \quad \text{when } \beta(n) = 0,\\
p_i(n+1) &= p_i(n) - \sum_{j \neq i} h_j(p(n)).
\end{aligned} \tag{17}$$

In the above $g_j, h_j$ are continuous nonnegative functions mapping $S_r \rightarrow [0, 1]$ further satisfying (for all $p \in S_r^0$)

$$\begin{aligned}
&0 < g_j(p) < p_j,\\
&0 < \sum_{j \neq i} [p_j + h_j(p)] < 1,
\end{aligned} \tag{18}$$

so that $p(n+1) \in S_r^0$ whenever $p(n) \in S_r^0$.

The following theorem gives conditions on $g_j, h_j$ for absolute expediency.

**Theorem 5.1** *A learning automaton using the general algorithm* (17) *is absolutely expedient if and only if*

$$g_1(p)/p_1 = g_2(p)/p_2 = \cdots = g_r(p)/p_r$$

*and*

$$h_1(p)/p_1 = h_2(p)/p_2 = \cdots = h_r(p)/p_r, \tag{19}$$

*for all* $p \in S_r^0$.

*Comments*: (1) The theorem says that an absolutely expedient scheme is determined by 2 functions only. These could be designated as $\lambda(p) = g_i(p)/p_i$ and $\mu(p) = h_i(p)/p_i$, ($i = 1, \ldots, r$). Reward–inaction schemes can be obtained by setting $\mu(p) \equiv 0$. The $L_{R-I}$ scheme results when $\lambda(p) \equiv a, \mu(p) \equiv 0$.

(2) When $d_m$ is unique, it can be shown that absolute expediency is equivalent to the

condition $\Delta p_m(n) > 0$ for all $n$, all $p \in S_r^0$ and in all stationary random environments*.
(3) Absolute expediency implies that $E[p_m(n)]$ is monotonically increasing in all stationary random environments*.
(4) Other classes of absolutely expedient schemes can be obtained by choosing different forms of reinforcement schemes. The most general scheme at present is due to Aso and Kimura (Narendra & Thathachar 1973).

### 5.5 *Estimator algorithms*

Estimator algorithms (Thathachar & Sastry 1985) arose from the idea that as the automaton is operating in the random environment, it is gaining information about the environment. This information can profitably be used in the updating of $p(n)$; for instance, to speed up the learning process. One such algorithm called the *pursuit algorithm* is given below.

Let

$$\hat{d}_i(n) = R_i(n)/Z_i(n), \tag{20}$$

where $R_i(n)$ = number of times reward input was obtained during the instants at which action $\alpha_i$ was selected up to the instant $n$.

$Z_i(n)$ = number of times action $\alpha_i$ was selected up to the instant $n$.
$\hat{d}_i(n)$ = estimate of $d_i$ at $n$.

Let

$$\hat{d}_H(n) = \operatorname*{Max}_i [\hat{d}_i(n)].$$

Then,

$$p(n+1) = p(n) + a[e_{H(n)} - p(n)], \tag{21}$$

where $0 < a < 1$ and $e_{H(n)}$ is the unit vector with unity in position $H(n)$ and zeros in the rest.

*Comments*: (1) The pursuit algorithm computes the 'optimal' vector $e_{H(n)}$ at each $n$ on the basis of the measurements up to $n$ and moves $p(n)$ towards it by a small distance determined by the parameter $a$.
(2) It can be shown that the pursuit algorithm is $\varepsilon$-optimal in all stationary random environments. Furthermore it is an order of magnitude faster than $L_{R-I}$ and other nonestimator algorithms.

## 6. Convergence and $\varepsilon$-optimality

The basic theoretical question in the operation of a learning automaton is the asymptotic behaviour of $\{p(n)\}$ with respect to $n$. It refers to the convergence of a sequence of dependent random variables.

There are two approaches to the analysis of the problem. One is based on stochastic contraction mapping principles leading to distance-diminishing operators. The other approach is through the martingale convergence theorem.

* Omit environments with all $d_i$ equal.

In the study of learning algorithms two distinct types of convergence can be identified. In the first type, information on the initial state $p(0)$ is eventually lost as $p(n)$ evolves in time. An algorithm with such behaviour is said to be ergodic. Here $p(n)$ converges in distribution to a random variable (r.v.) $p$ whose distribution is independent of $p(0)$. This is characteristic of algorithms such as $\mathrm{L}_{\mathrm{R-I}}$ and $\mathrm{L}_{\mathrm{R-\varepsilon P}}$.

In the second type of convergence, the process $\{p(n)\}$ has a finite number of absorbing states. It can be shown that $p(n)$ converges w.p.1 to one of the absorbing states. This type of convergence is associated with $\mathrm{L}_{\mathrm{R-I}}$ and other absolutely expedient schemes which are also called absorbing algorithms.

In order to show ε-optimality, one has to consider the effect of small values of the learning parameter. In ergodic algorithms, the problem can be reduced to the study of an associated ordinary differential equation which is shown to have a stable equilibrium point near the optimum value of $p(n)$. In absorbing algorithms, bounds on the probability of convergence to the optimum value of $p(n)$ are derived and it is shown that these bounds converge to 1 as the learning parameter goes to zero.

## 7. *Q* and *S* models

The development so far has been concerned with $P$ models where the environment has only a binary response. Most of these results can be extended to $Q$ and $S$ model environments. Actually it is enough to consider $S$ models as $Q$ models could be regarded as particular cases of $S$ models.

In $S$ models the output of the environment for each action $\alpha_i$ is a random variable with a distribution $F_i$ over the interval $[0, 1]$. The mean value of this distribution $s_i$ plays the same role as the reward probability $d_i$ in the $P$ model,

Let

$$s_i = E[\beta(n) \mid \alpha(n) = \alpha_i] \tag{22}$$

This $s_i$ is usually called the reward strength.

Each reinforcement scheme in the $P$ model has its counterpart in the $S$ model. For instance, the $\mathrm{L}_{\mathrm{R-I}}$ scheme can be extended as follows.

*The $SL_{R-I}$ scheme*

$$\begin{aligned} p_i(n+1) &= p_i(n) - a\beta(n)p_i(n), \quad \text{if } \alpha(n) \neq \alpha_i, \\ p_i(n+1) &= p_i(n) + a\beta(n)(1 - p_i(n)), \quad \text{if } \alpha(n) = \alpha_i. \end{aligned} \tag{23}$$

The scheme reduces to the $\mathrm{L}_{\mathrm{R-I}}$ scheme when $\beta(n) = 0$ or 1.

*Comment*: When the output of the environment is a continuous variable such as the performance index of a controlled process, an $S$ model has to be used. If $Y(n)$ is the output whose upper and lower bounds are $A$ and $B$, it can be transformed to lie in the interval $[0, 1]$ by defining

$$\beta(n) = [Y(n) - B]/[A - B].$$

However, $A$ and $B$ are not always known in practice and may have to be estimated. The $S$ model version of the pursuit algorithm avoids this problem as it needs only the average output due to each action up to the present instant $n$.

## 8. Hierarchical systems

When the number of actions is large, a single automaton becomes ineffective. It becomes slow as a large number of action probabilities are to be updated. One way of overcoming this complexity is to arrange a number of automata in a hierarchical system (Ramakrishnan 1982).

Figure 2 shows a number of automata arranged in 2 levels of a tree hierarchy. The first level automaton has $r$ actions and chooses an action (say $\alpha_i$) based on its action probability distribution. This action triggers the automaton $A_i$ of the second level which in turn chooses action $\alpha_{ij}$ based on its own distribution. The action $\alpha_{ij}$ interacts with the environment and elicits a response $\beta(n)$. The reward probability is $d_{ij}$. The action probabilities of both $A$ and $A_i$ are now updated and the cycle is repeated.

The basic problem here is to find reinforcement schemes for the automata at different levels which ensure the convergence of the hierarchy to the optimal action. We shall outline an approach which results in absolute expediency.

Let the following notation be used.

$\beta(n) =$ response of the environment at $n$.
$p_i(n) =$ $i$th action probability of automaton $A$ at $n$.
$p_{ij}(n) =$ $j$th action probability of automaton $A_i$ at $n$.
$d_{ij} =$ reward probability associated with action $\alpha_{ij}$ at the second level and $\alpha_i$ at the first level
$= P[\beta(n) = 1 | \alpha_i, \alpha_{ij}]$
$r_i =$ number of actions of $A_i$.

Let the first level automaton $A$ use a reward–inaction absolutely expedient scheme as follows.

$$\alpha_i = \text{action selected at } n.$$

$$\left.\begin{aligned} p_i(n+1) &= p_i(n) + \lambda(p(n))[1 - p_i(n)] \\ p_j(n+1) &= p_j(n)[1 - \lambda(p(n))] \end{aligned}\right\} \text{if } \beta(n) = 1$$
$$p_k(n+1) = p_k(n), \quad (i = 1, \ldots, r), \quad \text{if } \beta(n) = 0. \tag{24}$$

Similarly let the second level updating be as follows.

$$\alpha_{ij} = \text{action selected at } n.$$

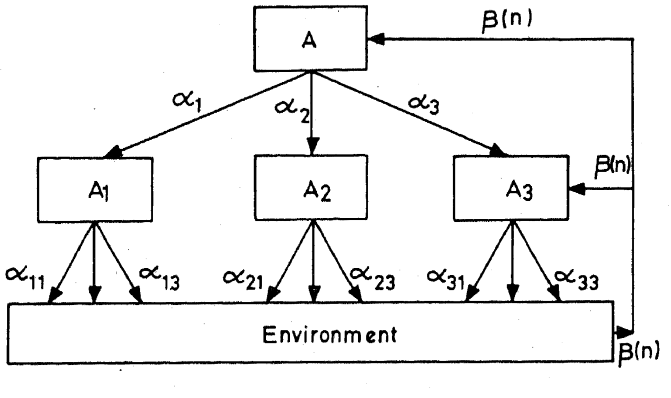

**Figure 2.** A 2-level hierarchical system.

*Automaton $A_i$*

$$p_{ij}(n+1) = p_{ij}(n) + \lambda_i(p(n))(1 - p_{ij}(n)), \quad \text{if } \beta(n) = 1,$$
$$p_{ik}(n+1) = p_{ik}(n)[1 - \lambda_i(p(n))], \quad (k \neq j). \tag{25}$$

*Automaton $A_k (k \neq i)$*

No updating

In terms of the above quantities there is a simple relationship which ensures absolute expediency of the hierarchy, i.e. the hierarchy is equivalent to a single automaton which is absolutely expedient.

**Theorem 8.1.** *The hierarchical system described by* (24), (25) *is absolutely expedient, if and only if*

$$\lambda_i(p(n)) = [\lambda(p(n))]/[p_i(n+1)] \tag{26}$$

*for all $n$, all $p(n) \in S_r^0$ and all $i = 1, \ldots, r$.*

*Comments*: (1) The results stated in theorem 8.1 can be extended to any number of levels. Basically one has to divide the $\lambda$ of each automaton by the connecting action probability of the previous level at $(n+1)$.
(2) The division operation need not cause concern as it will not lead to division by zero w.p.1.
(3) Number of updatings is reduced from $r^2$ to $2r$ in a 2-level hierarchy and $r^N$ to $Nr$ in an $N$-level hierarchy.

## 9. Team of learning automata

An alternative manner in which a complex learning problem can be handled is through a number of learning automata arranged to form a team. Each member of the team has the same goal. An example of such a team is in a game with identical payoffs. Here each automaton of the team $A_1, A_2, \ldots, A_N$ has a finite number of actions to choose from. At an instant $n$, each automaton chooses one action following which the environment gives out the same payoff to each member of the team. The automata update their action probabilities and choose new actions again. The process repeats. The objective of all the automata is to maximize the common payoff. The set up is shown in figure 3 (Ramakrishnan 1982).

The automata can operate in total ignorance of other automata, in which case it is a decentralized game. Alternatively some kind of information transfer can take place among the automata so as to improve the speed of operation or other characteristics of the game.

Let the action set of automaton $A_k$ be $\alpha^k$ and the action probability vector at instant $n$ be $p(k, n)$ $(k = 1, 2, \ldots, N)$. Also let each automaton operate according to a generalized nonlinear reward–inaction scheme described below.

Let the action chosen by $A_k$ at $n$ be $\alpha_{i_k}^k$. Then for each $k = 1, 2, \ldots, N$,

$$p(k, n+1) = p(k, n) + (G^k)^T e_{i_k}(k), \quad \text{for } \beta(n) = 1,$$
$$p(k, n+1) = p(k, n). \tag{27}$$

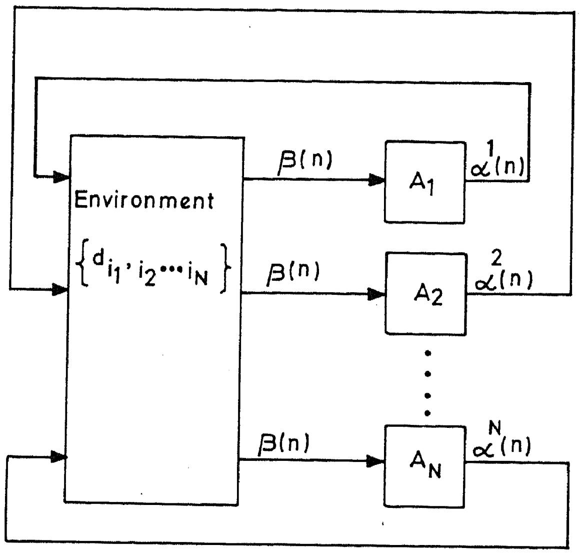

**Figure 3.** Automata game with common pay-off.

where
(i) $e_{ik}(k)$ is the $(r_k \times 1)$ unit vector having unity for the $i_k$th element and the rest of the elements are zero.
(ii) $G^k$ stands for $G^k\,(p(k,n))$ and $G^k(p)$ is an $r_k \times r_k$ matrix with its $(i,j)$th element $g^k_{ij}(p)$ $(j \neq i)$ and $(i,i)$th element $g^k_{ii}(p) = -\,\Sigma_{j \neq i} g^k_{ij}(p)$.

The environment is described by a hypermatrix of reward probabilities defined by

$$d_{i,i_2,\ldots,i_N} = P[\beta(n) = 1 \,|\, \alpha^k_{i_k} \text{ chosen by } A_k \text{ at } n,\ k = 1,\ldots,N]. \tag{28}$$

For optimality, each automaton $A_k$ should converge to the action $\alpha^k_{m_k}$ w.p.1, where

$$d_{m1m2}, \ldots, m_N = \max_{i,\ldots,i_N} [d_{i,i_2,\ldots,i_N}]. \tag{29}$$

A notion related to absolute expediency can be defined as follows.

DEFINITION 9.1

A learning algorithm for the game of $N$ automata with identical payoffs is said to be absolutely monotonic if

$$W(n) = E[W(n+1) - W(n) \,|\, p(k,n), k = 1,\ldots,N] > 0 \tag{30}$$

for all $n$, all $p(k,n) \in S^0_{r_k}$ and all possible game envirᴗnments.

*Comment:* Absolute monotonicity ensures monotonic increase of $E[W(n)]$ in arbitrary game environments.

The class of absolutely monotonic learning algorithms can be characterized as follows.

**Theorem 9.1.** *Necessary and sufficient conditions for the learning algorithm* (27) *to be absolutely monotonic are given by*

$$p^T(k,n) G^k(p(k,n)) = 0 \tag{31}$$

*for all* $k = 1,\ldots,N$, *all* $n$ *and all* $p(k,n) \in S^0_{r_k}$.

*Comment*: It can be checked that absolutely expedient algorithms of the reward–inaction type are also absolutely monotonic. Thus the $L_{R-I}$ scheme is absolutely monotonic. It appears difficult to include general penalty terms.

A team using an absolutely monotonic learning algorithm may not be ε-optimal. To appreciate the difficulty consider a 2-player game where each automaton player has 2-actions. The game environment can be represented by the matrix of reward probabilities $D$.

$$D = [d_{ij}] = \begin{bmatrix} 0{\cdot}8 & 0{\cdot}1 \\ 0{\cdot}3 & 0{\cdot}6 \end{bmatrix}.$$

Here the rows correspond to actions of automaton $A_1$ and columns to those of $A_2$. If both the automata choose the first action, the probability of getting a reward is 0·8.

Looking at the matrix $D$, one can identify 0·8 and 0·6 as local maxima as they are the maximum elements of their row and column. However, 0·8 is the global maximum and we would like the automata $A_1$ and $A_2$ to converge to their first actions as this leads to the highest expected payoff.

The problem with absolutely monotonic schemes is that they could converge to any of the local maxima. Only in the case of a single local maximum do they lead to ε-optimality.

In spite of the above limitation, the above result is of fundamental importance in the decentralized control of large systems. It indicates that simple policies used independently by an individual decision maker could lead to desirable group behaviour.

Estimator algorithms could be used to overcome the difficulty associated with a team converging to a local maximum. It can be shown that these algorithms converge to the global maximum in the sense of ε-optimality. The pursuit algorithm applied to the game with common payoff can be stated as follows (Thathachar & Sastry 1985; Mukhopadhyay & Thathachar 1989).

1. $A_1, A_2, \ldots, A_k, \ldots, A_N$ are the $N$ automata players;
2. the $k$th player $A_k$ has $r_k$ strategies (actions), $k = 1, 2, \ldots, N$;
3. $\{\alpha_1^k, \alpha_2^k, \ldots, \alpha_{r_k}^k\}$ is the set of actions of $A_k$;
4. $p(k,n) = [p_1(k,n), p_2(k,n), \ldots, p_r{}^k(k,n)]^T$ is the action probability vector of $A_k$ at instant $n$;
5. $\alpha^k(n) =$ action selected by $A_k$ at $n$.
6. $\beta(n) =$ payoff at $n$(0 or 1) common to all players;
7. $D$ is the $N$-dimensional hypermatrix with elements

$$d_{i_1, i_2, \ldots, i_N} = P[\beta(n) = 1 \mid \alpha^k(n) = \alpha_{i_k}^k,\ k = 1, \ldots, N];$$

8. $d_{m_1, m_2, \ldots, m_N} = \max_{i_1, i_2, \ldots, i_N} (d_{i_1, i_2, \ldots, i_N})$;
9. each $A_k$ maintains an estimate of $D$ in $\hat{D}(n)$ as indicated in the next algorithm;
10. $E^k = [E_1^k, E_2^k, \ldots, E_{r_k}^k]^T, \quad k = 1, \ldots, N,$

and

$$E_j^k = \max_{\substack{i_s,\, 1 \leqslant s \leqslant N \\ s \neq k}} \{d_{i_1, i_2, \ldots, i_{k-1} j i_{k+1}, \ldots, i_N}\}$$

$$\hat{E}_j^k(n) = \max_{\substack{i_s,\, i \leqslant s \leqslant N \\ s \neq k}} \hat{d}(n)_{i_1, i_2, \ldots, i_{k-1} j i_{k+1}, \ldots, i_N}\}$$

is an estimate of $E_j^k$ at $n$;

11. $H_k$ is a random index such that

$$\hat{E}^k_{H_k}(n) = \max_i \{\hat{E}^k_i(n)\}.$$

## 9.1 *The pursuit algorithm*

Let $\alpha^k(n) = \alpha^k_i$ $(k = 1, \ldots, N)$. The updating of $p(k, n)$ is as follows,

$$p(k, n+1) = p(k, n) + a[e_{H_k} - p(k, n)]. \tag{32}$$

In the above, $e_{H_k}$ is the unit vector with unity in the $H_k$th position and zero in the rest. The parameter $a$ satisfies $0 < a < 1$. The reward probability estimates are updated as follows.

$$\begin{aligned}
R_{i_1 i_2 \cdots i_N}(n+1) &= R_{i_1 i_2 \cdots i_N}(n) + \beta(n),\\
R_{j_1 j_2 \cdots j_N}(n+1) &= R_{j_1 j_2 \cdots j_N}(n), \quad (j_k \neq i_k),\\
Z_{i_1 i_2 \cdots i_N}(n+1) &= Z_{i_1 i_2 \cdots i_N}(n), \quad +1\\
Z_{j_1 j_2 \cdots j_N}(n+1) &= Z_{j_1 j_2 \cdots j_N}(n), \quad (j_k \neq i_k).
\end{aligned}$$

In the above $Z_{i_1 i_2 \cdots i_N}(n)$ represents the count of the number of times the action set $\{\alpha^1_{i_1}, \alpha^2_{i_2}, \ldots, \alpha^N_{i_N}\}$ has been selected up to the instant $n$. $R_{i_1 i_2 \cdots i_N}(n)$ is the number of times the reward was obtained at the instants the same action set was selected up to instant $n$. Thus the estimates of the reward probabilities can be computed as

$$\hat{d}_{j_1 j_2 \cdots j_N}(n+1) = [R_{j_1 j_2 \cdots j_N}(n)]/[Z_{j_1 j_2 \cdots j_N}(n)]. \tag{33}$$

The convergence result for the team using this algorithm can be stated as follows.

**Theorem 9.2** *In every stationary randon game environment, a team of learning automata playing a game with identical payoffs using the pursuit algorithm is ε-optimal. That is, given any* $\varepsilon > 0$, $\delta > 0$, *there exist* $a^* > 0$, $n_0 < \infty$ *such that*

$$P[|p_{m_k}(k, n) - 1| < \varepsilon] > 1 - \delta \tag{34}$$

*for all* $n > n_0, 0 < a < a^*$ *and* $k = 1, 2, \ldots, N$.

*Outline of Proof.* The proof of the theorem depends on 3 main ideas.

(1) If the learning parameter $a$ is chosen to be sufficiently small, all action $N$ tuples are selected any specified number of times with probability close to unity.
(2) Under the above conditions, $\hat{D}(n) \to D$ and $\hat{E}(n) \to E$.
(3) For each of the automata $A_k$ the game playing algorithm is equivalent to a single pursuit algorithm with $\hat{E}^k_j$ taking the role of the estimate of the reward probability. Hence each automaton converges to the optimal action $\alpha^k_{mk}$ in the sense indicated.

*Comment*: In a practical learning problem the parameter $a$ has to be chosen carefully. Too small a value will make learning too slow to be useful. Too large a value may speed up convergence but may also lead to convergence to the wrong actions.

## 10. Application to pattern recognition

Let patterns represented by $m$-dimensional vectors from a feature space $X$ belong to one of two classes $w_1$ and $w_2$. The class conditional densities $f(x/w_i)$ and the prior probabilities $P(w_i)$ $(i = 1, 2)$ are assumed to be unknown. Only a set of sample patterns with known classification is available for training the pattern recognizer (Thathachar & Sastry 1987).

To classify new patterns, a discriminant function $g(x): R^m \to R$ is defined such that the following rule can be used.

$$\text{If}\, g(x) \geqslant 0, \quad \text{decide } x \in w_1,$$
$$\text{if}\, g(x) < 0, \quad \text{decide } x \in w_2. \tag{35}$$

The task is to determine $g(x)$ which minimizes the probability of misclassification.

It is known that the Bayes' decision rule,

$$x \in w_1, \text{if}\, P(w_1/x) \geqslant P(w_2/x), \tag{36}$$

minimizes the probability of misclassification. Thus the optimum discriminant function is given by

$$g_{\text{opt}}(x) = P(w_1/x) - P(w_2/x), \tag{37}$$

and must be determined using the training samples.

Let a known form of discriminant function with $N$ parameters $\theta_1, \theta_2, \ldots, \theta_N$ be assumed.

$$g(x) = h(\theta_1, \theta_2, \ldots, \theta_N, x). \tag{38}$$

Then with the decision rule (35), the probability of correct classification for a given $\theta = (\theta_1, \ldots, \theta_N)$ is given by

$$J(\theta) = P(w_1)P(g(x) \geqslant 0 | x \in w_1) + P(w_2)P(g(x) < 0 | x \in w_2). \tag{39}$$

For a sample pattern $x$, let

$$L(x) = 1, \quad \text{if } x \text{ is properly classified,}$$
$$= 0, \quad \text{otherwise.} \tag{40}$$

Then,

$$J(\theta) = E[L(x)], \tag{41}$$

and hence maximizing $E[L(x)]$ maximizes the probability of correct classification.

The pattern classification problem can now be posed as a game of $N$ automata $A_1, A_2, \ldots, A_N$ each of which chooses a parameter $\theta_i$ $(i = 1, \ldots, N)$ to maximize the identical payoff $J(\theta)$. Hence the results of § 9 can be applied provided each parameter $\theta_i$ is discretized and consequently belongs to the finite action set of $A_i$. Further, since complete communication between the automata can be assumed, estimator algorithms can be used to speed up convergence rates.

The training of the classifier proceeds as follows. Let $\theta_i \in \alpha^i$ where $\alpha^i$ is the finite action set of $A_i$ defined by

$$\alpha^i = \{\alpha^i_1, \alpha^i_2, \ldots, \alpha^i_r\}.$$

In the game, each automaton $A_i$ chooses a particular action (and hence a value of $\theta_i$), and this results in a classifier with a discriminant function $g(x)$. Any sample pattern is classified according to rule (35) and $L(x)$ is determined according to (36). This $L(x)$ forms the payoff for all the automata that update their action probabilities following an estimator algorithm. If the set of actions selected at any stage is $\{\alpha^1_{i_1}, \alpha^2_{i_2}, \ldots, \alpha^N_{i_N}\}$ the corresponding reward probability is $d_{i_1, i_2, \ldots, i_N}$. For using estimator algorithms it is necessary to estimate $d_{i_1, i_2, \ldots, i_N}$ and update the estimate with each pattern. As estimator algorithms are $\varepsilon$-optimal, the parameter converges to its optimal value with arbitrary accuracy when a proper selection of the learning parameter is made in the algorithm.

Although $L_{R-I}$ and other absolutely monotonic algorithms can also be used for updating action probabilities, estimator algorithms are preferred for two reasons. The estimator algorithms invariably converge to the global optimum whereas the former converge only to local maxima. Furthermore the nonestimator algorithms are very slow and need substantially larger training sets.

In conclusion, the automata team learns the optimal classifier under the following assumptions:

(1) The form of the optimal discriminant function is contained in the functional form of $g(x)$ chosen.
(2) The optimal values of the parameters are elements of the sets $\alpha^i (i = 1, 2, \ldots, N)$.

Even when these assumptions are not satisfied, the team of automata would learn the best classifier among the set of classifiers being considered. By choosing finer parameter sets, successively better approximations to the optimal classifier can be obtained.

*Example*: Let $X = [x_1, x_2]^T$ be the feature vector with independent features. The class conditional densities

$$f(x_1 | w_1) \text{ and } f(x_2 | w_1) \text{ are uniform over } [1, 3],$$

$$f(x_1 | w_2) \text{ and } f(x_2 | w_2) \text{ are uniform over } [2, 4].$$

A discriminant function of the form

$$g(X) = 1 - (x_1/\theta_1) - (x_2/\theta_2),$$

is assumed with a range of parameters [0, 10]. This interval is discretized into 5 values.

The discriminant function learned by a 2-automaton team is

$$g(X) = 1 - (x_1/5) - (x_2/5),$$

which is optimal.

The average number of iterations for the optimum action probabilities to converge to 0·95 is 930. The team converged to the optimal discriminant function in 8 out of 10 runs. In the other two runs it converges to a line on the $x_1$–$x_2$ plane close to the optimal.

## 11. Other applications

There are several areas of computer and communication engineering where learning automata have been found to be significantly useful. A few of these are outlined below.

There has been a great deal of interest in applying adaptive routing to communication networks. Since these networks generally involve large investments, even a small improvement in their working efficiency results in appreciable savings in operational expenses. Inherently, communication networks are such that the volume as well as the pattern of traffic vary over wide ranges. Unusual conditions arise because of component failures and natural disasters. These considerations lead to the necessity of using learning control algorithms to achieve improved performance under uncertain conditions.

In circuit switched networks (such as telephone networks), a learning automaton is used at a node to route the incoming calls to other nodes (Narendra & Mars 1983). Typically if a call from a source node $i$, destined for node $j$ is at node $k$, an automaton $A_{ij}^k$ is used to route the calls at node $k$. The actions of the automaton are either the links connected to node $k$ or the sequence in which the connecting links are to be tried. The environmental response is the information whether the call reached the destination or not. Typically $\mathrm{L}_{\mathrm{R}-\varepsilon\mathrm{P}}$ and $\mathrm{L}_{\mathrm{R}-\mathrm{P}}$ schemes have been used in adaptive routing. It has been observed that when $\mathrm{L}_{\mathrm{R}-\varepsilon\mathrm{P}}$ schemes are used, the blocking probabilities along the route at each node are equalized. Similarly blocking rates are equalized when $\mathrm{L}_{\mathrm{R}-\mathrm{P}}$ schemes are used for routing.

It has been generally concluded that automata at the various nodes act in such a manner as to equalize the quality of service at the different nodes as measured by blocking probabilities corresponding to different loads. In the presence of abnormal operating conditions such as link failure, the automata algorithms result in significantly reduced blocking probabilities and node congestion provided additional capacity is available in the network.

Similar remarks apply to the use of automata for routing messages in packet-switched networks also (Mason & Gu 1986, pp. 213–228).

There are also some queueing problems such as task scheduling in computer networks where automata play a useful role. Learning automata have been shown to provide a good solution to the image data compression problem by selecting a proper code to transmit an image over a communication channel (Narendra & Thathachar 1989). Another prominent application is in the consistent labelling problem (Thathachar & Sastry 1986) where a team of automata provide consistent labels to objects in a scene. A consequence of this is an efficient solution of low-level vision problems such as stereocorrespondence and shape matching (Sastry *et al* 1988; Banerjee 1989).

## 12. Generalized learning automata

Collectives of learning automata such as hierarchies and teams have been seen to be effective in handling complex learning problems such as pattern recognition and also in a number of other applications. These are examples of certain types of interconnection of a number of automata and could be regarded as artificial neural networks where the learning automaton plays the role of an artificial neuron. The terminology seems appropriate because of the parallel and distributed nature of the interconnection and the perceptual tasks the collectives can perform. However, the nature of interconnection is somewhat different from that usually assumed in the literature, where the output of each unit forms one of the inputs to several other units. The only input to the learning automaton is the response of the environment and there is no other input from other automata. Thus a generalization of the automaton appears necessary to provide the type of interconnection normally envisaged.

The structure of a learning automaton can be generalized in two directions (Williams 1988). The first generalization is in parametrization of the state space of the automaton. Instead of the simplex $S_r$, one could have an arbitrary state space $\theta$ and use a mapping $\theta \rightarrow S_r$ to generate the action probabilities. For instance, in the case of a two-action automaton whose state space $\theta$ is the real line, one could use the mapping

$$\psi(\theta) = 1/(1 + e^{-\theta}) = p_1$$

to compute $p_1$, the probability of selection of $\alpha_1$. Functions suitable for higher dimensions are harder to identify.

The second generalization one could consider is to allow the automata to have another input apart from the response of the environment. The idea is that the optimal action of the automaton may differ in differing contexts. The second input is meant for representing the context and is called the context input or context vector. When the context input is a constant, the structure of the automaton reduces to the classical one.

A parametrized-state learning automaton with context input is called a generalized learning automaton (Narendra & Thathachar 1989). It has also been called an associative stochastic learning automaton (William 1988) as it is trying to learn which actions are to be associated with which context inputs. One could think of the decision space divided into a number of regions where each region is associated with a context input and the task of the automaton is to determine the best action associated with each context input.

Generalized learning automata can be connected together to form networks which interact with the environment. Here the actions of individual automata are treated as outputs and these may in turn serve as context inputs to other automata or as outputs of the network. Some automata may receive context inputs from the environment. The environment also produces the usual response (also called reinforcement signal) which is common to all the automata and is a globally available signal. In this respect the network shares a property with the game with identical payoff. A schematic diagram of such a network is shown in figure 4.

The following operations take place in the network over a cycle.

1. The environment picks an input pattern (i.e. set of context inputs) for the network randomly. The distribution of the inputs is assumed to be independent of prior events within the network or environment.
2. As the input pattern to each automaton becomes available, it picks an action randomly according to the action probability distribution corresponding to that particular input. This 'activation' of automata passes through the network towards the 'output side'.
3. After all the automata at the output side of the network have selected their actions, the environment picks a reinforcement signal randomly according to a distribution corresponding to the particular network output pattern chosen and the particular context input to the network.
4. Each automaton changes its internal state according to some specified function of its current state, the action just chosen, its context input and the reinforcement signal.

While the above concept of a network of generalized learning automata is conceptually simple and intuitively appealing, there are very few analytical results

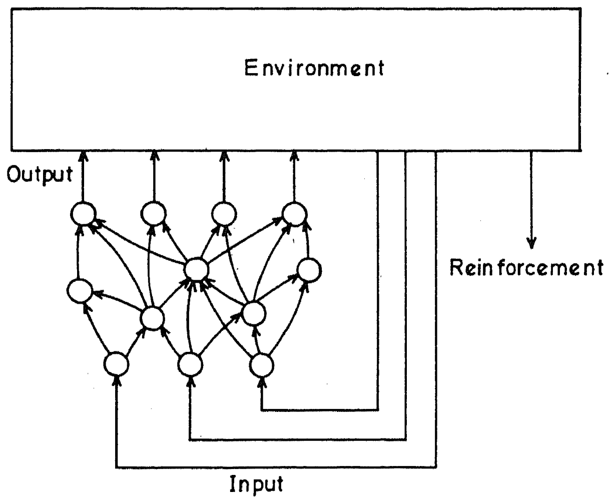

**Figure 4.** Network of generalized learning automata.

concerning it. Even the available results are not as complete as those of hierarchies or games of learning automata. New techniques may be needed for the analysis of such networks and for bringing out the tasks for which they are well suited.

## References

Banerjee S 1989 *On stochastic relaxation paradigms for computational vision*, PhD thesis, Electrical Engineering Department, Indian Institute of Science, Bangalore

Bush R R, Mosteller F 1958 *Stochastic models for learning* (New York: John Wiley and Sons)

Lakshmivarahan S 1981 *Learning algorithms – theory and applications* (New York: Springer Verlag)

Lakshmivarahan S, Thathachar M A L 1973 Absolutely expedient learning algorithms for stochastic automata. *IEEE Trans. Syst., Man Cybern.* SMC-3: 281–283

Mason L G, Gu X D 1986 Learning automata models for adaptive flow control in packet-switching networks. In *Adaptive and learning systems* (ed.) K S Narendra (New York: Plenum)

Mukhopadhyay S, Thathachar M A L 1989 Associative learning of Boolean functions. *IEEE Trans. Syst. Man Cybern.* SMC-19: 1008–1015

Narendra K S, Mars P 1983 The use of learning algorithms in telephone traffic routing – A methodology, *Automatica* 19: 495–502

Narendra K S, Thathachar M A L 1989 *Learning automata – an introduction* (Englewood Cliffs, NJ: Prentice Hall)

Ramakrishnan K R 1982 *Hierarchical systems and cooperative games of learning automata*, PhD thesis, Indian Inst. Sci. Bangalore

Sastry P S, Banerjee S, Ramakrishnan K R 1988 A local cooperative processing model for low level vision. *Indo-US Workshop on Systems and Signal Processing* (ed.) N Viswanadham (New Delhi: Oxford & IBH)

Shapiro I J, Narendra K S 1969 Use of stochastic automata for parameter self-optimization with multimodal performance criteria. *IEEE Trans. Syst. Sci. Cybern.* SSC-5: 352–360

Thathachar M A L, Sastry P S 1985 A new approach to the design of reinforcement schemes for learning automata. *IEEE Trans. Syst., Man Cybern.* SMC-15: 168–175

Thathachar M A L, Sastry P S 1986 Relaxation labeling with learning automata. *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-8: 256–268

Thathachar M A L, Sastry P S 1987 Learning optimal discriminant functions through a cooperative game of automata. *IEEE Trans. Syst., Man Cybern.* SMC-17: 73–85

Tsetlin M L 1973 *Automaton theory and modeling of biological systems* (New York: Academic Press)

Williams R J 1988 Toward a theory of reinforcement – learning connectionist systems, Technical Report, NU-CCS-88-3, College of Computer Science, Northeastern University, Boston, MA