TRACKING CHANGING STOCHASTIC MODELS VIA STOCHASTIC BINARY NEURAL NETWORKS

Anthony Burrell*
e-mail: tburrell@coe.eng.ua.edu.
Achilles Kogiantis**
P. Papantoni-Kazakos*
e-mail: tpapanto@coe.eng.ua.edu.

*Electrical Engineering Dept., 317 Houser Hall, Box 870286, University of Alabama, Tuscaloosa, AL 35487-0286
**Electrical & Computer Engineering Dept., Univ. of Southwestern Louisiana, P.O. Box 43890, Lafayette, LA 70504-3890

Abstract

We consider a sequential algorithm that detects changes from an acting stochastic model to any one of a number of alternatives. We adopt discrete approximations of the stochastic models and we propose the deployment of stochastic binary neural networks which are pretrained to produce the appropriate statistical measures associated with these models. The pretraining is implemented by a backpropagating supervised learning algorithm of stochastic approximation nature which converges almost surely under general conditions. The overall system performance is discussed and some numerical results are presented.

I. INTRODUCTION

In this paper, we consider the neural implementation of a sequential algorithm for the detection of change in the acting stochastic environmental model, when the model alternatives must be learned themselves in interaction with the environment. The original algorithm for parametrically designed stochastic models can be found in Bansal et al [1] and its parametric extension can be found in Burrell [3] and Burrell et al [4], [5]. The applications of the algorithm are numerous, ranging from quality control, to the detection of edges in images, to the recognition of failures in network lines, to the dynamic capacity allocation in multimedia ATM networks. For the latter application, see Burrell [3] and Burrell et al [4].

II. THE ALGORITHM AND DISCRETE ROBUST MAPPINGS

The extended parametric problem in Burrell [3] and Burrell et al [4], [5] is as follows.

Let x_1^n denote the sequence $x_1, ..., x_n$ of n observations after time zero. Let the process which initially generates the data sequence be the process μ_0. Let it be possible that a shift to any one of m-1 independent processes μ_i ; i=1, ..., m-1 may occur at any point in time, where if a $\mu_0 \rightarrow \mu_i$ shift occurs, then the process μ_i remains active thereafter. The objective is to detect the occurrence of a $\mu_0 \rightarrow \mu_i$ shift as accurately and as timely as possible, including the detection of the process μ_i which μ_0 changed to. Let us denote by f_i ; i = 0, ..., m-1 density or probability functions induced by the processes μ_i ; i =1, ..., m-1 and let $f_i(x_n|x_1^{n-1})$ denote the density/probability function at x_n, conditioned on the sequence x_1^{n-1}. Then, the parametric algorithm in [3] is as follows:

(a) Select a threshold $\delta_0 > 0$.

(b) Have m-1 parallel algorithms operating. The ith algorithm; i =1, ..., m-1 is monitoring a $\mu_0 \rightarrow \mu_i$ shift. $T_n^{0i}(x_1^n)$ denotes the operating value of the ith algorithm at time n, given the observation sequence x_1^n. The operating value $T_n^{0i}(x_1^n)$ is updated as follows.

$$T_0^{0i} \equiv 0$$

$$T_n^{0i}(x_1^n) = \max\left(0,\ T_{n-1}^{0i}(x_1^{n-1}) + \log \frac{f_i(x_n|x_1^{n-1})}{f_0(x_n|x_1^{n-1})}\right) \quad (1)$$

(c) The algorithmic system stops the first time n when either one of the m-1 parallel algorithms crosses the common threshold δ_0. If the ith algorithm is the one that first crosses the threshold, then it is declared that a $\mu_0 \rightarrow \mu_i$ shift has occurred.

Let N_{0i} denote the extended stopping variable induced by the ith algorithm in the system; that is, $N_{0i} \triangleq inf\left\{n: T_n^{0i}(x_1^n) \geq \delta_0\right\}$. Let us define the following quantities.

$$L_n^{ij}(x_1^n) \triangleq n^{-1} \log \frac{f_j(x_1^n)}{f_i(x_1^n)} \qquad ;i, j = 0, 1, .., m-1$$

$$I_{ij} \triangleq \lim_{n\rightarrow\infty} L_n^{ij}(x_1^n) \qquad ;i, j = 0, 1, .., m-1 \quad (2)$$

$$p_n^{ij}(v) \triangleq P\left(L_n^{ij}(x_1^n) < v \,|\, \mu_j\right) \qquad ;i, j = 0, 1, .., m-1$$

Consider then, the following conditions:

(A) $I_{ij}; i,j = 0,1,..,m-1$ exist and

$$I_{ij} = E\{I_{ij} | \mu_j\}, \text{ a.s. } (P_{\mu_j})$$

(B) For $i, j = 0, 1, .., m-1$ and for $v \in (0, I_{ij})$,

$$\lim_{n\to\infty} n\, p_n^{ij}(v) = 0 \text{ and } \sum_{n\geq 1} p_n^{ij}(v) < \infty$$

Conditions (A) state the existence of the generalized Kullback-Leibler information numbers $\{I_{ij}\}$ while conditions (B) ensure the convergence of the large-deviation probabilities $\{p_n^{ij}(v)\}$ fast enough. The following theorem and corollary can now be expressed, whose proof can be found in Burrell et al [5].

Theorem 1

Let the processes $\{\mu_j; j = 0, 1, .., m-1\}$ be stationary, ergodic, mutually independent, and satisfying conditions (A) and (B). Then,

$$\text{As } \delta_0 \to \infty, \quad \underset{\substack{i=1,..,m-1}}{\underset{j=0,1,..,m-1}{E\{N_{0i} | \mu_j\}:}}$$

$$: \begin{cases} \sim [I_{0j} - I_{ij}]^{-1} \log \delta_0 & ; \text{ if } I_{0j} \geq I_{ij} \\ \geq 2^{-1}\delta_0 & ; \text{ if } I_{0j} < I_{ij} \end{cases} \quad (3)$$

Corollary

Given that the process $\mu_j; j=1, .., m-1$ is acting throughout, the algorithmic system will asymptotically detect the $\mu_0 \to \mu_j$ shift correctly, in the expected stopping time sense. That is:

$$E\{N_{0j} | \mu_j\} \sim I_{0j}^{-1} \log \delta_0:$$

$$\begin{cases} < E\{N_{0i} | \mu_j\} \sim [I_{0j} - I_{ij}]^{-1} \log \delta_0; \forall i \neq j: I_{0j} > I_{ij} \\ < E\{N_{0i} | \mu_j\} \geq 2^{-1}\delta_0 \quad ; \forall i \neq j: I_{0j} < I_{ij} \end{cases} \quad (4)$$

Given that the process $\mu_j; j=1, .., m-1$ is acting throughout, the asymptotic expected stopping time of the algorithmic system is: $E\{N_{0j} | \mu_j\} \sim I_{0j}^{-1} \log \delta_0$.

Remarks: (a) The algorithm is asymptotically optimal in the sense that, as $\delta_0 \to \infty$ and for any extended stopping variables $\{N'_{0j}; j = 1, ..., m-1\}$, such that $E\{N'_{0j} | \mu_0\} \geq 2^{-1}\delta_0$, then $E\{N_{0j} | \mu_j\} \leq E\{N'_{0j} | \mu_j\}$. (b) From Theorem 1 and its corollary, the importance of the generalized Kullback-Leibler information numbers $\{I_{ij}\}$ is clear. The larger these numbers are and the closer to each other, the better is the performance of the extended algorithm, in the sense that the smaller are then the ratios

$$E\{N_{0j} | \mu_j\} / E\{N_{0i} | \mu_j\}; \quad ; i \neq j;\ i,j = 1,..., m-1; \text{as } \delta_0 \to \infty.$$

(c) The asymptotic correct detections induced by the extended algorithm, as stated by the corollary, are due to the single common threshold δ_0 used. The latter is basically associated with the starting process μ_0.

The implicit assumption in the algorithm in (1) is that the conditional densities/probabilities $\{f_i(x_n | x_1^{n-1})\}$ are analytically known. In reality, these densities/probabilities may have to be estimated and/or may not possess analytic forms. In addition, it is generally desirable that the environment be mapped into a set of distinct representations for computability. We thus propose the deployment of m neural networks that are *pretrained* to reproduce discrete versions of the conditional probabilities/densities $\{f_i(x_n | x_1^n)\}$. The latter networks will then provide the appropriate inputs to the algorithm in (1).

Discrete Finite Memory Robust Mappings

Let $x_1, ..., x_n$ denote a sequence of observations, representing the environment. Then, given the ith environmental model, given $x_1, ..., x_n$, the objective of the discrete mapping is to predict which one of M distinct regions, the observation x_{n+1} is going to be in. Denoting these regions $A_j; j=1, ..., M$, a high-performance predictive encoding operation requires in fact the computation of the conditional probabilities, $\{p_{ij}(x_1, ..., x_n) \triangleq P(x_{n+1} \varepsilon A_j / x_1, ..., x_n)\}_{1\leq j\leq M}$; which are used to map an observed sequence $\{x_1, ..., x_n\}$ onto each of the regions $\{A_j\}$, with corresponding probabilities $\{p_{ij}(x_1, ..., x_n)\}$. Two problems arise immediately:

(i) Exploding computational load, due to the increasing memory represented by the sequences $(x_1, ..., x_n)$.

(ii) Statistical information on the sequences $(x_1, ..., x_n)$ needed for the computation of the probabilities $\{p_{ij}(x_1, ..., x_n)\}$.

The first problem is resolved, if the increasing memory is approximated by finite, say size-*l* memory. That is, the increasing computational load is, instead, bounded if the process that generates the observations is approximated by an *l*-order Markov process. Then, the information loss is minimized when the process is Gaussian (see Blahut [2]).

Thus, to reduce the exploding computational load due to increasing data memory, we may *initially* model

the process that generates the environmental data or observations by an *l*-order Gaussian Markov process, whose autocorrelation mxm matrix Q_i, has components identical to the corresponding components of the original process. We name this initial (Gaussian and Markov) process, *nominal* process.

Starting with our nominal process, but incorporating then statistical uncertainties in our model, we are led to a powerful (qualitatively) robust formalization, which results in a stochastic mapping (see Papantoni-Kazakos et al [8]), as follows:

Given observations (x_1, ..., x_n), use the *l* most recent observations for the prediction of the next datum x_{n+1}, and defining $y_m^T = [x_{n-m+1}, \ldots, x_n]$, decide that $x_{n+1} \varepsilon A_j$ with probability $q_{ij}(y_m)$, defined as follows,

$$q_{ij}(y_m) = \frac{1}{M} r_i(y_m) + [1 - r_i(y_m)] p_{ij}(y_m), \qquad (5)$$

where $p_{ij}(y_m)$ is the conditional probability of $x_{n+1} \varepsilon A_j$, given $y_m^T = [x_{n-m+1}, \ldots, x_n]$, under the ith model, as induced by the Gaussian and Markov nominal process, and where, for some positive finite constant λ_i,

$$r_i(y_m) \triangleq 1 - \min\left[1, \frac{\lambda_i}{\sqrt{y_m^T Q_i^{-1} y_m}}\right] \qquad (6)$$

The value of the constant λ_i in (6) represents the level of confidence to the "purity" of the data vector y_m, in terms of it being generated by the nominal Gaussian process: the higher the value of λ_i, the higher the level of confidence, where as λ_i decreases, increased weight on purely random mappings (represented by the probability $\frac{1}{M}$ per region) is induced.

In addition, robust estimation of the autocorrelation matrix Q_i will be required. The components of the autocorrelation matrix Q_i should emerge from the statistics of the nominal Gaussian process. It is thus necessary to provide a scheme for the robust estimation of the matrix Q, in which observations generated from the outlier process are rejected in the estimation of the components of the matrix, (see Kazakos et al [6]). Special attention should be paid to allow for the existence of the inverse autocorrelation matrix Q_i^{-1} from the estimates of its components.

The robust prediction expression in (5) was based on a Gaussian assumption for the nominal process that generates the data in the environment, where the latter assumption was the result of an information-theoretical approach to the reduction of the computational load caused by increased past memory. The important robust effects induced by the mapping in (5) remain unaltered, however, when instead, the probability $p_{ij}(y_m)$ in (5) arises from an arbitrary nonGaussian process, and when its conditioning on y_m is substituted by conditioning on quantized values of the scalar quantity $y_m^T Q_i^{-1} y_m$. When quantized values are involved, the implementation of the mapping in (5) involves the following stages:

(a) *Preprocessing*. This stage corresponds to long-term memory and involves the robust preestimation, (see Kazakos et al [6]), and storage of the matrix Q_i^{-1}.

(b) *Processing*. This stage corresponds to short-term memory. It uses the matrix Q_i^{-1} from the preprocessing step and the observation vector y_m to: (i) first compute the quadratic expression $y_m^T Q_i^{-1} y_m$, (ii) then, represent $y_m^T Q_i^{-1} y_m$ in a quantized form comprised of N distinct values, (iii) finally, use the quantized values in (ii) to compute the corresponding value of the function $r_i(y_m)$ in (6).

(c) *Predictive Mapping*. This stage involves the estimation of the probabilities $\{p_{ij}(y_m)\}$ and the computation of the probabilities $\{q_{ij}(y_m)\}$ in (5) using inputs from the processing stage, and the subsequent implementation of the prediction mappings.

The three different stages above are performed sequentially by separate but connected neural structures, named *preprocessing layer*, *processing layer*, and *predictive mapping layer*, respectively. Our focus in this paper is on the latter layer: its structure and its operations. Towards that direction, we first note that, due to the quantization operations at the processing layer, the expression in (5) takes the following form:

$$\begin{aligned} q_{ij\rho} &= \frac{1}{M} r_{i\rho} + [1 - r_{i\rho}] p_{ij\rho}\ ; \\ &; \text{for } y_m^T Q_i^{-1} y_m \to R_\rho\ ;\ \rho = 1, \ldots, N \end{aligned} \qquad (7)$$

where $q_{ij\rho}$, $p_{ij\rho}$, and $r_{i\rho}$ denote, respectively, the probabilities $q_{ij}(y_m)$ and $p_{ij}(y_m)$ and the number $r_i(y_m)$, when the quantized value of $y_m^T Q_i^{-1} y_m$ equals R_ρ.

III. THE PREDICTIVE MAPPING LAYER

Consider the integer M in (7), and let s be a unique positive integer, such that $2^{s-1} < M \leq 2^s$. Then, in modulo-2 arithmetic, each state *j*; *j*=1, ..., M can be represented by an s-length 0-1 binary sequence $x_1 \cdots x_s$. The state R_ρ is provided as an input to the prediction layer by the

processing layer, and the former produces a binary sequence $x_1 \cdots x_s$ as a prediction mapping. Given the state R_ρ, the operations of the prediction layer must be such that, a given prediction sequence $x_1 \cdots x_s$ is produced stochastically with probability.

$$q_i(x_1 \cdots x_s / R_\rho) = \frac{1}{M} r_{i\rho} + [1 - r_{i\rho}] p_i(x_1 \cdots x_s / R_\rho) \quad (8)$$

where expression (8) is the same as expression (7); when the binary representation of the positive integer j in the latter is $x_1 \cdots x_s$, and where $p_i(x_1 \cdots x_s / R_\rho)$ is the prediction mapping generated by the nominal process that represents the *actual data environment*. Due to the stochastic nature of the rule in (8), such is also the nature of the predictive mapping layer, whose neural representation corresponds then to a stochastic neural network, first developed by Kogiantis et al [7], when the response of each neuron is limited to binary. We proceed with the description of the latter representation.

Let us temporarily assume that the probabilities $p_i(x_1 \cdots x_s / R_\rho)$ have been "learned" and are known. Without lack in generality, let us also assume that $M=2^s$. The original constraint of binary firing per neuron in the prediction layer leads us to the digital representation of the future states, $\{x_1, \ldots, x_s\}$. The design can be accommodated easily in a binary tree structure. In detail, given the observed state R_ρ and the resulting $R_{i\rho}$ value, the mapping $x_1 \cdots x_s$ can be obtained via a stochastic binary tree search, on the 2^s-leaves tree, as follows: (a) With probability $r_{i\rho}$ a fair tree-search is activated, where the tree-node x_1; $x_1=0, 1$ is visited with probability 0.5, and each of the two tree-nodes branching off a visited tree-node $x_1 \cdots x_k$; $1 \le k \le s-1$ is also visited with probability 0.5. (b) With probability $1-r_{i\rho}$ a generally biased tree-search is activated, where the tree-node x_1 is visited with probability $p_i(x_1/R_\rho)$, while from a visited tree-node $x_1 \cdots x_k$; $1 \le k \le s-1$ the tree-node $x_1 \cdots x_k x_{k+1}$ is visited with probability:

$$p_i(x_{k+1} / x_1 \cdots x_k, R_\rho) \triangleq p_i(x_1 \cdots x_k x_{k+1} / x_1 \cdots x_k, R_\rho)$$

where,

$$\begin{aligned} p_i(x_1 \cdots x_s / R_\rho) &= p_i(x_1 / R_\rho) \cdots \\ & p_i(x_{k+1} / x_1 \cdots x_k, R_\rho) \cdots p_i(x_s / x_1 \cdots x_{s-1}, R_\rho) \end{aligned} \quad (9)$$

Thus, the predictive mapping layer may be viewed as been comprised of a fair-search binary tree and a number of biased-search binary trees, each of the latter corresponding to a specific observation state R_ρ. Given R_ρ, the common fair-search binary tree is activated with probability $r_{i\rho}$, while, with probability $1-r_{i\rho}$, the biased-search binary tree that corresponds to the state R_ρ is activated, instead; we name the latter tree, the R_ρ tree.

Given the observation state R_ρ, consider the R_ρ -tree in conjunction with the sequential stochastic representation in (9) of the corresponding prediction mappings, as generated by the process representing the actual data environment. Let $u_{x_1 \cdots x_k}; 1 \le k \le s$ represent the binary random output of the neuron that corresponds to the node $x_1 \cdots x_k$ of the R_ρ -tree. Then, $u_{x_1 \cdots x_k} = 1$ if and only if $u_{x_1 \cdots x_i} = 1; \forall i \le k$. Thus, the output $u_{x_1 \cdots x_k}$ may be viewed as generated by a product, $W_{x_1} W_{x_2/x_1} \cdots W_{x_k/x_1 \cdots x_{k-1}}$, of mutually independent binary random variables $\{W_{x_i/x_1 \cdots x_{i-1}}\}_{1 \le i \le k}$, whose distributions at the operational stage of the R_ρ -tree must be as follows, (in view of (9)):

$$\begin{aligned} P(u_{x_1 \cdots x_k} = 1) &= P(W_{x_1} W_{x_2/x_1} \cdots W_{x_k/x_1 \cdots x_{k-1}} = 1) = \\ &= p_i(x_1 \cdots x_k / R_\rho) = p_i(x_1 / R_\rho) \cdots \\ & p_i(x_k / x_1 \cdots x_{k-1}, R_\rho) = \\ &= P(W_{x_1} = 1)\, P(W_{x_2/x_1} = 1) \ldots \\ & P(W_{x_k/x_1 \cdots x_{k-1}} = 1); \; 2 \le k \le s \\ P(u_{x_1} = 1) &= P(W_{x_1} = 1) = p_i(x_1 / R_\rho), \end{aligned} \quad (10)$$

where,

$$\begin{aligned} & P(W_{x_k/x_1 \cdots x_{k-1}} = 1) = \\ & = p_i(x_k / x_1 \cdots x_{k-1}, R_\rho); \; 2 \le k \le s \end{aligned} \quad (11)$$

The above logical arguments and expressions lead to the following neural structure of the R_ρ-tree: (a) The neuron corresponding to the tree-node x_1; $x_1=0, 1$ has a binary random variable W_{x_1} built in, where $W_0=1-W_1$. At the operational stage, the neuron must be activated with probability $p_i(x_1/R_\rho)$; thus, $P(W_{x_1} = 1) = p_i(x_1 / R_\rho)$ then, where $P(W_1 = 1) = 1 - P(W_0 = 1)$. (b) For $k \ge 2$, the neuron corresponding to the tree-node $x_1 \cdots x_k$ has a binary random variable $W_{x_k/x_1 \cdots x_{k-1}}$ built in and fires, if and only if the latter variable takes the value 1 and simultaneously the neuron corresponding to the tree-node $x_1 \cdots x_{k-1}$ fires as well. Thus, the binary neural output $u_{x_1 \cdots x_k}$ is formed as a product $u_{x_1 \cdots x_{k-1}} W_{x_k/x_1 \cdots x_{k-1}}$, where,

$$P(u_{x_1\cdots x_k}=1) = P(u_{x_1\cdots x_{k-1}} W_{x_k/x_1\cdots x_{k-1}}=1) = \\ = P(u_{x_1\cdots x_{k-1}}=1)P(W_{x_k/x_1\cdots x_{k-1}}=1) \quad (12)$$

and where, at the operational stage of the R_ρ -tree, the probability $P(W_{x_k/x_1\cdots x_{k-1}}=1)$ must be as in (11). We note that,

$$W_{1/x_1\cdots x_{k-1}} = 1 - W_{0/x_1\cdots x_{k-1}};\ \forall k \geq 2;\ \forall x_1\cdots x_{k-1} \quad (13)$$

and thus,

$$P(W_{1/x_1\cdots x_{k-1}}=1) = \\ = 1 - P(W_{0/x_1\cdots x_{k-1}}=1);\ \forall k \geq 2;\ \forall x_1\cdots x_{k-1}$$

Learning at the Predictive Mapping Layer.

Given the R_ρ -tree, we observe that, due to (10), any adaptations of the probability $P(u_{x_1\cdots x_s}=1)$ *backpropagate* to adaptations of each of the probabilities $P(W_{x_1}=1)$, …, $P(W_{x_s/x_1\cdots x_{s-1}}=1)$. It thus suffices to focus on the learning of the probabilities, $\{P(u_{x_1\cdots x_s}=1)\}$, for the various binary sequences $x_1\cdots x_s$, which correspond to the responses of the output or "visible" neurons in the R_ρ -tree network. For easiness in presentation, let us now consider a fixed sequence $x_1\cdots x_s$, (in conjunction with the fixed observed state R_ρ that represents the R_ρ -tree). Let then p denote the value of the probability $p(x_1\cdots x_s / R_\rho)$, as induced by the environment, and let q denote the value of the probability $P(u_{x_1\cdots x_s}=1)$. Let the natural number n denote discrete observation time from the beginning of the learning stage, and let $\hat{p}_n$ and $\hat{q}_n$ denote estimates at time n of the probability values p and q, respectively. Finally, let the random variable V_n be defined as equal to 1; if the environmental event $\{x_1\cdots x_s / R_\rho\}$ occurs at time n, and as equal to 0; otherwise, and let

$Z_\rho \triangleq V_\rho(1-W_\rho)$, $W_\rho = \begin{cases} 0;\ \text{w.p. } 1-r_{i\rho} \\ 1;\ \text{w.p. } 1-r_{i\rho} \end{cases}$. In Kogiantis et al [7], a Kullback-Leibler matching criterion between p and q was used, in conjunction with Newton's iterative numerical method, to develop the supervised learning algorithm stated below.

ALGORITHM

Initial Values: Select an initial value $\hat{q}_1 > 0$, while $\hat{p}_1 = v_1$.

Computational Steps:

(a) Given computed value $\hat{p}_n$ and z_{n+1}, as in (17), compute $\hat{p}_{n+1}$ as follows:

$$\hat{p}_{n+1} = \hat{p}_n + \frac{z_{n+1}[1-r_{i\rho}]^{-1} - \hat{p}_n}{n+1} \quad (14)$$

For some small positive value δ, the value $\hat{p}_{n+1}$ is corrected to δ; if $\hat{p}_{n+1} < \delta$, and is corrected to 1-δ; if $\hat{p}_{n+1} > 1-\delta$.

(b) Given computed values $(\hat{q}_n, \hat{p}_n)$, given z_{n+1}, compute $\hat{q}_{n+1}$ as follows:

$$\hat{q}_{n+1} = \hat{q}_n - \frac{(\hat{q}_n - \hat{p}_n)\hat{q}_n(1-\hat{q}_n)}{(\hat{q}_n - \hat{p}_n)^2 + \hat{p}_n(1-\hat{p}_n)} + \\ + \frac{z_{n+1}[1-r_{i\rho}]^{-1} - \hat{p}_n}{n+1}\left[\frac{\hat{q}_n(1-\hat{q}_n)}{(\hat{q}_n - \hat{p}_n)^2 + \hat{p}_n(1-\hat{p}_n)} + \right]^2 \quad (15)$$

where,

$\hat{p}_{n+1} - \hat{p}_n = \frac{z_{n+1}[1-r_{i\rho}]^{-1} - \hat{p}_n}{n+1}$, from (14).

For some small positive value ζ, the value $\hat{q}_{n+1}$ is corrected to ζ; if $\hat{q}_{n+1} < \zeta$, and is corrected to 1-ζ; if $\hat{q}_{n+1} > 1-\zeta$.

In Kogiantis et al [7], the following theorem was proved.

Theorem 2

Let the process which generates the observed data in the environment be ergodic. Let then, s denote the probability of the event $\{x_1\cdots x_s / R_\rho\}$, as induced by the latter process. Then, the supervised learning algorithm converges to the probability s, almost surely, with rate inversely proportional to the sample/iteration size n.

In Kogiantis et al [7], it was found that the learning algorithm converges rapidly to predictive probability

mappings that are close to those induced by the environment, even under mismatch network conditions. Specifically, when past dependence decays fast with distance, then, even when the network order is less than the order of the Markovian environmental model, convergence to almost the true process is attained in less than fifty iterations, in most cases.

IV. CONCLUSIONS

We considered a sequential algorithm for the detection of change from a given environmental model to a number of alternatives, when the latter models need to be learned via supervised environmental observations. We assumed discretized observations and stochastic neural networks for the supervised learning of the models. We also adopted a backpropagating supervised learning algorithm for the pretraining of these networks, that guarantees almost sure and rapid convergence under general model conditions. The overall system is efficient and accurate, as well as robust, and has numerous applications.

REFERENCES

[1] R.K. Bansal and .P. Papantoni-Kazakos, "An Algorithm for Detecting a Change in a Stochastic Process," *IEEE Trans. Inf. Th.*, Vol. IT-32, pp. 227-235, March 1986.

[2] R.E. Blahut, "Hypothesis Testing and Information Theory," *IEEE Trans. Inf. Th.*, Vol. IT-20, pp. 405-417, 1987.

[3] A. Burrell, "Traffic Management in Broadband Integrated Services Digital Networks," Ph.D. Dissertation, Univ. of Virginia, Aug. 1994

[4] A. Burrell, D. Makrakis, and P. Papantoni-Kazakos, "Traffic Monitoring for Capacity Allocation of Multimedia Traffic in Broadband Networks," in review.

[5] A. Burrell and P. Papantoni-Kazakos, "Extended Sequential Algorithm for Detecting Changes in Acting Stochastic Processes," in review.

[6] D. Kazakos and P. Papantoni-Kazakos, Detection and Estimation, Computer Science Press, 1989,

[7] A.G. Kogiantis and P. Papantoni-Kazakos, "Operations and Learning in Neural Networks for Robust Prediction," *IEEE Trans. on Systems, Man, and Cybernetics*, June 1996, Vol. 27, Part B.

[8] Papantoni-Kazakos, D. Kazakos, and K. Birimwal, "Predictive Analog-to-Digital Conversion for Resistance to Data Outliers," *Information and Computation*, March, 1992, Vol. 97, No. 1, 1992.