Int. J. Communications, Network and System Sciences, **2012, 5, 603-608**
http://dx.doi.org/10.4236/ijcns.2012.529070 Published Online September 2012 (http://www.SciRP.org/journal/ijcns)

Stochastic Binary Neural Networks for Qualitatively Robust Predictive Model Mapping

A. T. Burrell[1], P. Papantoni-Kazakos[2]
[1]Computer Science Department, Oklahoma State University, Stillwater, USA
[2]Electrical Engineering Department, University of Colorado Denver, Denver, USA
Email: tburrell@okstate.edu, Titsa.Papantoni@ucdenver.edu

Received May 15, 2012; revised June 25, 2012; accepted July 10, 2012

ABSTRACT

We consider qualitatively robust predictive mappings of stochastic environmental models, where protection against outlier data is incorporated. We utilize digital representations of the models and deploy stochastic binary neural networks that are pre-trained to produce such mappings. The pre-training is implemented by a back propagating supervised learning algorithm which converges almost surely to the probabilities induced by the environment, under general ergodicity conditions.

Keywords: Qualitative Robustness; Predictive Model Mapping; Stochastic Approximation; Stochastic Binary Neural Networks; Real-Time Supervised Learning; Ergodicity

1. Introduction

We consider the case where the statistical behavior of environmental models must be learned in real time. In particular, we focus on learning such behavior predictively, as may be applicable in data compression, hypothesis testing or model identification, while statistical qualitative robustness for protection against outlier data is sought as well. In this paper, we promote the deployment of stochastic binary neural networks which implement predictive model mappings in real time, in interaction with the environment; *i.e.* supervised learning, while they also offer sound protection against data outliers. Our approach uses results from stochastic approximation and statistical qualitative robustness [1-9]. While powerful such results have been in existence for a long time, they have not been given attention synergistically, in the light of neural network implementations. In this paper, our objective is to stimulate interest in the application of the existing theories in such implementations, especially those addressing environmental models.

When neural networks operate in stochastically described environments, supervised learning corresponds to a statistical sequential estimation problem dealt with by stochastic approximation methods. There is rich literature in such methods represented by the works of Abdelhamid [1], Beran [10], Blum [11], Fabian [2], Fisher [12], Gerencser [13], Kashyab *et al.* [14,15], Kiefer *et al.* [4], Kushner [5], Kushner *et al.* [6], Ljung [7], Ljung *et al.* [8], Robbins *et al.* [16] and Young *et al.* [9].

In the neural networks literature, supervised learning has been basically limited to techniques arising from the Robbins/Monro [16] method and its extensions, with performance criterion the least squares error. The representative works on the subject are those by Barron *et al.* [17], Elman *et al.* [18], Gorman *et al.* [19], Minsky *et al.* [20], Rosenblatt [21], Werbos [22], White [23], Widrow [24], and Widrow *et al.* [25]. Literature in the area, when the performance criterion is, instead, the Kullback-Leibler distance (see Blahut [26] and Kazakos *et al.* [3]) and the techniques used do not necessarily arise from the Robbins/Monro method, is represented by the works of Ackley *et al.* [27], Amari *et al.* [28], Pados *et al.* [29-33] and Kogiantis *et al.* [34].

In the domain of stochastic neural networks, some more recent results address time-delay issues (Liu *et al.* [35] and Wang *et al.* [36]), while the book by Ling [37] discusses some general aspects in this area.

The organization of this paper is as follows. In Section 2, we introduce digital finite memory qualitatively robust predictive mappings, as well as the neural network layers needed for their implementation. In Section 3, we describe the operations performed at the predictive neural network layer. In Section 4, we present the supervised learning algorithm used at the predictive layer. In Section 5, we draw some conclusions.

2. Digital Finite Memory Qualitatively Robust Mapping

We consider digital environmental representations. We start by letting $x_1,\cdots,x_n$ denote a sequence of discrete-time observations that represent the environment. Then, given $x_1, \cdots, x_n$, the objective of the digital mapping is to predict which one of M distinct regions, the observation x_{n+1} is going to lie in. Denoting these regions A_j; $j = 1, \cdots, M$, let us define the probabilities $\left\{p_j(x_1, \cdots, x_n) \triangleq P\left(x_{n+1} \varepsilon A_j / x_1, \cdots, x_n\right)\right\}_{1 \le j \le M}$, which are used to map stochastically an observed sequence $\{x_1, \cdots, x_n\}$ onto each of the regions $\{A_j\}$, with corresponding probabilities $\{p_j(x_1, \cdots, x_n)\}$. Two problems arise immediately:

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

2) Statistical information on the sequences $(x_1, \cdots, x_n)$ needed for the computation of the probabilities $\{p_j(x_1, \cdots, x_n)\}$.

The first problem is resolved if the increasing memory is approximated by finite, say size-*m* memory. That is, the increasing computational load is, instead, bounded if the process that generates the observations is approximated by an *m*-order Markov process. Then, the information loss is minimized when the process is Gaussian (see Blahut [26]).

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 *m*-order Gaussian Markov process, whose auto-covariance $m \times m$ matrix Q has components identical to those 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 the form of unknown data outliers, we are led to a powerful qualitatively robust formalization, which results in a stochastic mapping (see Papantoni-Kazakos *et al.* [38]), as follows:

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

$$q_j(y_m) = \frac{1}{M} r(y_m) + \left[1 - r(y_m)\right] p_j(y_m), \qquad (1)$$

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

$$r(y_m) \triangleq 1 - \min\left[1, \frac{\lambda}{\sqrt{y_m^T Q^{-1} y_m}}\right] \qquad (2)$$

The value of the constant λ in (2) represents the level of confidence on 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, the higher the level of confidence, where as λ decreases, increased weight on purely random mappings (represented by the probability $\frac{1}{M}$ per region) is induced.

Robust estimation of the auto-covariance matrix *Q* may be also required. The components of the auto-covariance matrix *Q* should emerge from the statistics of the nominal Gaussian process. A scheme for the robust estimation of the matrix *Q* may arise from robust parameter estimation techniques, (see Kazakos *et al.* [3]).

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

1) *Preprocessing*. This stage corresponds to long-term memory and involves the robust pre-estimation (see Kazakos *et al.* [3]), and storage of the matrix Q^{-1}.

2) *Processing*. This stage corresponds to short-term memory. It uses the matrix Q^{-1} from the preprocessing step and the observation vector y_m to: a) first compute the quadratic expression $y_m^T Q^{-1} y_m$, b) subsequently represent $y_m^T Q^{-1} y_m$ in a quantized form comprised of *N* distinct values and c) finally, use the quantized values in d) to compute the corresponding value of the function *r* (y_m) in (2).

3) *Predictive Mapping*. This stage involves the estimation of the probabilities $\{p_j(y_m)\}$ and the computation of the probabilities $\{q_j(y_m)\}$ in (1) 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 (1) takes the following form:

$$\begin{aligned} q_{j\rho} &= \frac{1}{M} r_\rho + \left[1 - r_\rho\right] p_{j\rho}; \\ &\text{for } y_m^T Q^{-1} y_m \to R_\rho\,;\ \rho = 1, \cdots, N \end{aligned} \qquad (3)$$

where $q_{j\rho}$, $p_{j\rho}$, and r_ρ denote, respectively, the probabilities $q_j(y_m)$ and $p_j(y_m)$ and the number $r(y_m)$, when the quantized value of $y_m^T Q^{-1} y_m$ equals R_ρ.

3. The Neural Predictive Layer

Consider the integer *M* in (3), and let s be a unique positive integer, such that $2^{s-1} < M \le 2^s$. Then, in modulo-2 arithmetic, each state *j*, $j = 1, \cdots, M$ 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.

$$q\left(x_1 \cdots x_s / R_\rho\right) = \frac{1}{M} r_\rho + \left[1 - r_\rho\right] p\left(x_1 \cdots x_s / R_\rho\right) \qquad (4)$$

where expression (4) is the same as expression (3) when the binary representation of the positive integer j in the latter is $x_1 \cdots x_s$, and where $p\left(x_1 \cdots x_s / R_\rho\right)$ is the prediction mapping generated by the nominal process that represents the *actual data generating environment*. Due to the stochastic nature of the rule in (4), 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.* [17], 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 $p\left(x_1 \cdots x_s / R_\rho\right)$ 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 $\{x_1, \cdots, x_s\}$. The design can be accommodated easily in a binary tree structure. In detail, given the observed state R_ρ and the resulting R_ρ 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: 1) With probability r_ρ 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; 2) With probability $1 - r_\rho$ a generally biased tree-search is activated, where the tree-node x_1 is visited with probability $p\left(x_1 / R_\rho\right)$, 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:

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

where

$$\begin{aligned} p\left(x_1 \cdots x_s / R_\rho\right) &= p\left(x_1 / R_\rho\right) \cdots p\left(x_{k+1} / x_1 \cdots x_k, R_\rho\right) \\ &\quad \cdots p\left(x_s / x_1 \cdots x_{s-1}, R_\rho\right) \end{aligned} \qquad (5)$$

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_ρ, while, with probability $1 - r_\rho$, the biased-search binary tree that corresponds to the state R_ρ is activated, instead; we name the latter tree, the R_ρ tree. The nodes of each of the above binary trees are neurons that "fire," if the corresponding tree-nodes are "visited." Given R_ρ, a specific mapping $x_1 \cdots x_s$ is generated either equiprobably from the fair-search binary tree with probability r_ρ, or from the R_ρ-tree via the sequential stochastic representation in (5) with probability $1 - r_\rho$. It is thus in the R_ρ-tree that the probabilities which generate the data of the environment must be "learned" and then used to generate prediction mappings.

Given the observation state R_ρ, consider the R_ρ-tree in conjunction with the sequential stochastic representation in (5) of the corresponding prediction mappings, as generated by the process representing the actual environmental data. 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 $\left\{W_{x_i / x_1 \cdots x_{i-1}}\right\}_{1 \le i \le k}$, whose distributions at the operational stage of the R_ρ-tree must be as follows (in view of (5)):

$$\begin{aligned} P\left(u_{x_1 \cdots x_k} = 1\right) &= P\left(W_{x_1} W_{x_2/x_1} \cdots W_{x_k / x_1 \cdots x_{k-1}} = 1\right) \\ &= p\left(x_1 \cdots x_k / R_\rho\right) = p\left(x_1 / R_\rho\right) \cdots p\left(x_k / x_1 \cdots x_{k-1}, R_\rho\right) \\ &= P\left(W_{x_1} = 1\right) P\left(W_{x_2/x_1} = 1\right) \ldots P\left(W_{x_k / x_1 \cdots x_{k-1}} = 1\right); \; 2 \le k \le s \\ P\left(u_{x_1} = 1\right) &= P\left(W_{x_1} = 1\right) = p\left(x_1 / R_\rho\right), \end{aligned} \qquad (6)$$

where

$$P\left(W_{x_k / x_1 \cdots x_{k-1}} = 1\right) = p\left(x_k / x_1 \cdots x_{k-1}, R_\rho\right); \; 2 \le k \le s \qquad (7)$$

The above logical arguments and expressions lead to the following neural structure of the R_ρ-tree: 1) 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 $p\left(x_1 / R_\rho\right)$; thus, $P\left(W_{x_1} = 1\right) = p\left(x_1 / R_\rho\right)$ then, where $P\left(W_1 = 1\right) = 1 - P\left(W_0 = 1\right)$; 2) 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

$$P\left(u_{x_1\cdots x_k}=1\right)=P\left(u_{x_1\cdots x_{k-1}}W_{x_k/x_1\cdots x_{k-1}}=1\right)$$
$$=P\left(u_{x_1\cdots x_{k-1}}=1\right)P\left(W_{x_k/x_1\cdots x_{k-1}}=1\right) \quad (8)$$

and where, at the operational stage of the R_ρ-tree, the probability $P\left(W_{x_k/x_1\cdots x_{k-1}}=1\right)$ must be as in (7). We note that

$$W_{1/x_1\cdots x_{k-1}}=1-W_{0/x_1\cdots x_{k-1}}, \forall k\ge 2,\ \forall x_1\cdots x_{k-1} \quad (9)$$

and thus

$$P\left(W_{1/x_1\cdots x_{k-1}}=1\right)=1-P\left(W_{0/x_1\cdots x_{k-1}}=1\right),$$
$$\forall k\ge 2,\ \forall x_1\cdots x_{k-1}$$

As it is clear from the derivations and arguments in this section, the parameters of interest in the R_ρ-tree neural network consist of the independent binary random variables W_1 and

$$\left\{W_{1/x_1\cdots x_{k-1}}; x_i=0,1; 1\le i\le k-1\right\}_{2\le k\le s},$$

whose distributions $p\left(1/R_\rho\right)$ and

$$\left\{p\left(1/x_1\cdots x_{k-1},R_\rho\right);\ x_i=0,1;\ 1\le i\le k-1\right\}_{2\le k\le s},$$

must be "learned" in advance, via interaction with the environment.

4. Learning at the Predictive Layer

Given the R_ρ-tree, we observe that, due to (6), any adaptations of the probability $P\left(u_{x_1\cdots x_s}=1\right)$ *back-propagate* to adaptations of each of the other involved probabilities. It thus suffices to focus on the learning of the probabilities $\left\{P\left(u_{x_1\cdots x_s}=1\right)\right\}$ 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 $p\left(x_1\cdots x_s/R_\rho\right)$, as induced by the environment, and let *q* denote the value of the probability $P\left(u_{x_1\cdots x_s}=1\right)$. 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 $\left\{x_1\cdots x_s/R_\rho\right\}$ occurs at time *n*, and as equal to 0, otherwise, and let

$$Z_\rho \triangleq V_\rho\left(1-W_\rho\right), W_\rho=\begin{cases}0;\ w\cdot p\cdot\left(1-r_\rho\right)\\ 1;\ w\cdot p\cdot r_\rho\end{cases}.$$

In Kogiantis *et al.* [17], 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:

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

$$\hat{p}_{n+1}=\hat{p}_n+\frac{z_{n+1}\left[1-r_\rho\right]^{-1}-\hat{p}_n}{n+1} \quad (10)$$

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 $1-\delta$ if $\hat{p}_{n+1}>1-\delta$.

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

$$\hat{q}_{n+1}=\hat{q}_n-\frac{\left(\hat{q}_n-\hat{p}_n\right)\hat{q}_n\left(1-\hat{q}_n\right)}{\left(\hat{q}_n-\hat{p}_n\right)^2+\hat{p}_n\left(1-\hat{p}_n\right)}$$
$$+\frac{z_{n+1}\left[1-r_{i\rho}\right]^{-1}-\hat{p}_n}{n+1}\left[\frac{\hat{q}_n\left(1-\hat{q}_n\right)}{\left(\hat{q}_n-\hat{p}_n\right)^2+\hat{p}_n\left(1-\hat{p}_n\right)}\right]^2 \quad (11)$$

where $\hat{p}_{n+1}-\hat{p}_n=\dfrac{z_{n+1}\left[1-r_\rho\right]^{-1}-\hat{p}_n}{n+1}$ from (10).

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 $1-\zeta$ if $\hat{q}_{n+1}>1-\zeta$.

Remarks: 1) Expression (10) is the computationally efficient recursive estimation format of the probabilities that represent the environment. 2) The expression in (11) includes correction terms induced by the Newton's iterative numerical method, when the latter is applied on the Kullback-Leibler matching criterion between environmental probabilities and probabilistic adaptations used in the supervised learning process. The last term in (11) converges to zero, as the estimate in (10) converges to its true value. The second term in (11) converges to zero as the estimate of *q* converges to the estimate of *p*. 3) The small δ and ζ correction biases are used to prevent the corresponding probability estimates from diverging to the 0 or 1 degenerate values.

We now proceed with the statement of a theorem first proved in Kogiantis *et al.* [34].

Theorem

Let the process which generates the observed data in the environment be ergodic. Let then *s* denote the probability of the event $\left\{x_1\cdots x_s/R_\rho\right\}$, 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*.

Proof Outline

Here, we present an outline of the Theorem's proof. 1) If the observed data are generated by an ergodic process, then the recursive sequential estimate in (10) will converge to the probability s. 2) The pair sequence $(\hat{q}_n, \hat{p}_n)$ is a two dimensional Markov process. The expected value of the drift $\hat{q}_{n+1} - \hat{q}_n$, conditioned on $\hat{q}_n = \hat{p}_n = s$ equals zero, as deduced from expression (11). 3) In view of the result in 2), it is then shown that the supremum of the conditional expected drift in 2) multiplied by $\hat{q}_n - s$ converges to negative values, for all values of the absolute difference $|\hat{q}_n - s|$ that are larger than some given positive small value. 4) Using finally Blum's condition [11], the results in 2) and 3) above guarantee almost sure convergence.

We note that in the Theorem, if the process that generates the observed data in the environment is ergodic, and if $\left\{s\left(x_1 \cdots x_s / R_\rho\right)\right\}$ denote the prediction mappings induced by the latter process, then, via the learning algorithm and with almost sure convergence, the prediction mappings produced by the predictive mapping layer are governed by the probabilities

$$\left\{q'\left(x_1 \cdots x_s / R_\rho\right) \triangleq \left(1 - r_\rho\right) s\left(x_1 \cdots x_s / R_\rho\right) + M^{-1} r_\rho\right\}$$

In Kogiantis *et al.* [34], 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.

5. Conclusion

We presented a neural network implementation for a digital qualitatively robust predictive mapping of environmental models. The mapping uses synergistically results from statistical qualitative robustness and stochastic binary neural representation to realize digital real-time predictive operations which identify the environmental model, while they simultaneously protect the operations against data outliers. The supervised learning algorithm recommended for the training of the neural network is based on stochastic approximation principles applied to the Kullback-Leibler matching criterion, in conjunction with Newton's iterative numerical method, and converges almost surely for models generated by ergodic processes. The considered predictive mappings have numerous applications, ranging from data compression, to model identification to sequential model hypothesis testing.

REFERENCES

[1] F. Abdelhamid, "Transformation of Observations in Stochastic Approximation," *Annals of Statistics*, Vol. 1, No. 6, 1973, pp. 1158-1174. doi:10.1214/aos/1176342564

[2] V. Fabian, "On Asymptotic Normality in Stochastic Approximation," *Annals of Mathematical Statistics*, Vol. 39, No. 4, 1968, pp. 1327-1332. doi:10.1214/aoms/1177698258

[3] D. Kazakos and P. Papantoni-Kazakos, "Detection and Estimation," Computer Science Press, New York, 1989.

[4] J. Kiefer and J. Wolfowitz, "Stochastic Estimation of the Maximum of a Regression Function," *Annals of Mathematical Statistics*, Vol. 23, No. 3, 1952, pp. 462-466. doi:10.1214/aoms/1177729392

[5] H. Kushner, "Asymptotic Global Behavior for Stochastic Approximations and Diffusions with Slowly Decreasing Noise Effects: Global Minimization via Monte Carlo," *SIAM Journal of Applied Mathematics*, Vol. 47, No. 1, 1987, pp. 169-185. doi:10.1137/0147010

[6] H. Kushner and D. Clark, "Stochastic Approximation Methods for Constrained and Unconstrained Systems," Springer-Verlag, Berlin, 1978. doi:10.1007/978-1-4684-9352-8

[7] L. Ljung, "Analysis of Recursive Stochastic Algorithms," *IEEE Transactions on Automatic Control*, Vol. 22, No. 4, 1977, pp. 551-575. doi:10.1109/TAC.1977.1101561

[8] L. Ljung and T. Söderström, "Theory and Practice of Recursive Identification," MIT Press, Cambridge, 1983.

[9] T. Y. Young and R. A. Westerberg, "Stochastic Approximation with a Nonstationary Regression Function," *IEEE Transactions on Information Theory*, Vol. IT-18, No. 4, 1972, pp. 518-519. doi:10.1109/TIT.1972.1054851

[10] R. Beran, "Adaptive Autoregressive Process," *Annals of the Institute of Statistical Mathematics*, Vol. 28, No. 1, 1976, pp. 77-89. doi:10.1007/BF02504731

[11] J. R. Blum, "Multidimensional Stochastic Approximation Procedure," *Annals of Mathematical Statistics*, Vol. 22, No. 4, 1954, pp. 737-744. doi:10.1214/aoms/1177728659

[12] R. A. Fisher, "The Goodness of Fit of Regression Formulae and the Distribution of Regression Coefficients," *Journal of the Royal Statistical Society*, Vol. 85, No. 4, 1922, pp. 597-612. doi:10.2307/2341124

[13] L. Gerencser, "Parameter Tracing of Time-Varying Continuous-Time Linear Stochastic Systems," In: C. I. Byrnes and A. Lindquist, Eds., *Modeling, Identification and Robust Controls*, North-Holland, Amsterdam, 1986, pp. 581-594.

[14] R. L. Kashyap and C. C. Blaydon, "Recovery of Functions from Noisy Measurements Taken at Randomly Selected Points and Its Application to Pattern Classification," *Proceedings of the IEEE*, Vol. 54, No. 8, 1966, pp. 1127-1129. doi:10.1109/PROC.1966.5051

[15] R. L. Kashyap, C. Blaydon and K. S. Fu, "Stochastic Approximation," In: J. M. Mendel and K. S. Fu, Eds., *Adaptive Learning and Pattern Recognition Systems*, Academic Press, New York, 1970, pp. 329-355.

doi:10.1016/S0076-5392(08)60499-3

[16] H. Robbins and S. Monro, “A Stochastic Approximation Method,” *Annals of Mathematical Statistics*, Vol. 22, No. 3, 1951, pp. 400-407. doi:10.1214/aoms/1177729586

[17] A. R. Barron, F. W. van Straten and R. L. Barron, “Adaptive Learning Network Approach to Weather Forecasting: A Summary,” *Proceedings of the International Conference on Cybernetics and Society*, 1977, pp. 724-727.

[18] J. Elman and D. Zipser, “Learning the Hidden Structure of Speech,” *Journal of the Acoustical Society of America*, Vol. 83, No. 4, 1988, pp. 1615-1626. doi:10.1121/1.395916

[19] P. Gorman and T. Sejnowski, “Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets,” *Neural Networks*, Vol. 1, No. 1, 1988, pp. 75-90. doi:10.1016/0893-6080(88)90023-8

[20] M. Minsky and S. Papert, “Perceptrons,” MIT Press, Cambridge, 1969.

[21] F. Rosenblatt, “The Perceptron: A Perceiving and Recognizing Automaton,” Report 85-60-1, Cornell Aeronautical Laboratory, Buffalo, New York, 1957.

[22] P. Werbos, “Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences,” Ph.D. Dissertation, Harvard University, Cambridge, 1974.

[23] H. White, “Some Asymptotic Results for Learning in Single Hidden-Layer Feedforward Network Models,” *American Statistical Association*, Vol. 84, No. 408, 1989, pp. 1003-1013. doi:10.1080/01621459.1989.10478865

[24] B. Widrow, “Generalization and Information Storage in Networks of Adaline Neurons,” In: M. D. Yovits, G. T. Jacobi and G. D. Goldstein, Eds., *Self-Organizing Systems*, Spartan Books, Washington DC, 1962, pp. 435-461.

[25] B. Widrow and M. E. Hoff, “Adaptive Switching Circuits,” 1960 IRE WESCON Convention Record, 1960, pp. 96-104.

[26] R. E. Blahut, “Hypothesis Testing and Information Theory,” *IEEE Transactions on Information Theory*, Vol. IT-20, 1987, pp. 405-417.

[27] D. H. Ackley, G. E. Hinton and T. J. Sejnowski, “A Learning Algorithm for Boltzman Machines,” *Cognitive Science*, Vol. 9, No. 1, 1985, pp. 147-169. doi:10.1207/s15516709cog0901_7

[28] S. Amari, K. Kurata and H. Nagoaka, “Information Geometry of Boltzman Machines,” *IEEE Transactions on Neural Networks*, Vol. 3, No. 2, 1992, pp. 260-271. doi:10.1109/72.125867

[29] D. Pados and P. Papantoni-Kazakos, “New Non-Least-Squares Neural Network Learning Algorithms for Hypothesis Testing,” *IEEE Transactions on Neural Networks*, Vol. 6, No. 3, 1995, pp. 596-609. doi:10.1109/72.377966

[30] D. Pados, K. W. Halford, D. Kazakos and P. Papantoni-Kazakos, “Distributed Binary Hypothesis Testing with Feedback,” *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 25, No. 1, 1995, pp. 21-42.

[31] D. Pados, P. Papantoni-Kazakos, D. Kazakos and A. Kogiantis, “On-Line Threshold Learning for Neyman-Pearson Distributed Detection,” *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 24, No. 10, 1994, pp. 1519- 1531. doi:10.1109/21.310534

[32] D. Pados and P. Papantoni-Kazakos, “A Note on the Estimation of the Generalization Error and the Prevention of Overfitting,” *IEEE International Conference on Neural Networks*, Orlando, 1994.

[33] D. Pados and P. Papantoni-Kazakos, “A Class of Neyman-Pearson and Bayes Learning Algorithms for Neural Classification,” *IEEE International Symposium on Information Theory*, Trondheim, 1-27 July 1994.

[34] A. G. Kogiantis and P. Papantoni-Kazakos, “Operations and Learning in Neural Networks for Robust Prediction,” *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 27, No. 3, 1997, pp. 402-411.

[35] Y. Liu, Z. Wang and X. Liu, “Robust Stability of Discrete-Time Stochastic Neural Networks with Time-Varying Delays,” 4*th International Symposium on Neural Networks*, Vol. 71, No. 4-6, 2008, pp. 823-833.

[36] Z. Wang, Y. Liu, M. Li and X. Liu, “Stability for Stochastic Cohen-Grossberg Neural Networks with Mixed Time Delays,” *IEEE Transactions on Neural Networks*, Vol. 17, No. 3, 2006, pp. 814-820. doi:10.1109/TNN.2006.872355

[37] H. Ling, “Stochastic Neural Networks,” LAP Lambert Academic Publishing, 2010.

[38] P. Papantoni-Kazakos, D. Kazakos and K. Birmiwal, “Predictive Analog-to-Digital Conversion for Resistance to Data Outliers,” *Information and Computation*, Vol. 98, No. 1, 1992, pp. 56-98. doi:10.1016/0890-5401(92)90042-E